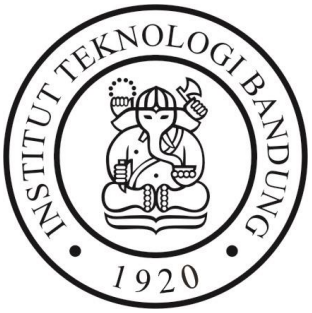
PETUNJUK PRAKTIKUM

SISTEM MIKROPROSESOR



Laboratorium Dasar Teknik Elektro

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

PETUNJUK PRAKTIKUM
SISTEM MIKROPROSESOR

Waskita Adijarto

Sandra Irawan

Triani Wulandari

Rhesa Aditya S.

Wilfrid Azariah

Ryan Dharma Chandra

Rizky Ardi Maulana

Akmal Narendra Sakti

Andi Muhammad Riyadhus Ilmy

Frans Jason

David Khowanto

Zefanya Chandra

Fauzan Rozin

M. Heronan Hyanda

Kelvin Sutirta

Gotawa Aryo Prakoso

Laboratorium Dasar Teknik Elektro

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

Semester II 2023 – 2024

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Aturan Umum Laboratorium Dasar Teknik Elektro

Kelengkapan

Setiap praktikan wajib berpakaian lengkap, mengenakan celana panjang/rok, kemeja dan mengenakan sepatu. Untuk memasuki ruang laboratorium praktikan wajib membawa kelengkapan berikut:

1. Modul praktikum,
2. Buku Catatan Laboratorium (BCL),
3. Alat tulis dan kalkulator,
4. Kartu Nama (Name tag), dan
5. Kartu Praktikum

Persiapan/ Sebelum Praktikum

Sebelum mengikuti percobaan sesuai jadwalnya, sebelum memasuki laboratorium praktikan harus mempersiapkan diri dengan melakukan hal-hal berikut:

1. Membaca dan memahami isi modul praktikum,
2. Mengerjakan hal-hal yang harus dikerjakan sebelum praktikum dilaksanakan, misalnya mengerjakan tugas pendahuluan, melakukan perhitungan-perhitungan, menyalin source code, mengisi Kartu Praktikum dsb.,
3. Mengisi daftar hadir di Tata Usaha Laboratorium,
4. Mengambil kunci loker dan melengkapi administrasi peminjaman kunci loker dengan kartu identitas (KTM/ SIM/ KTP).

Selama Praktikum

1. Setelah dipersilahkan masuk dan menempati bangku dan meja kerja, praktikan haruslah:
2. Menuliskan identitas diri pada Berita Acara Praktikum yang diedarkan oleh asisten,
3. Memperhatikan dan mengerjakan setiap percobaan dengan waktu sebaik-baiknya, diawali dengan kehadiran praktikan secara tepat waktu,
4. Mengumpulkan Kartu Praktikum pada asisten,
5. Melakukan pengecekan terhadap peralatan praktikum (termasuk kabel di dalam boks kabel) sebelum memulai praktikum,
6. Mendokumentasikan dalam Buku Catatan Laboratorium. (lihat Petunjuk Penggunaan BCL) tentang hal-hal penting terkait percobaan yang sedang dilakukan.

Setelah Praktikum

Setelah menyelesaikan percobaan, praktikan harus

1. Memastikan BCL dan Kartu Praktikum telah ditandatangani oleh asisten,
2. Mengembalikan kunci loker dan melengkapi administrasi pengembalian kunci loker
3. (pastikan kartu identitas KTM/ SIM/ KTP diperoleh kembali),
4. Mengerjakan laporan dalam bentuk SoftCopy (lihat Panduan Penyusunan Laporan di laman http://ldte.stei.itb.ac.id),
5. Mengumpulkan file laporan dengan cara mengunggah di laman http://praktikum.stei.itb.ac.id. Waktu pengiriman paling lambat jam 16.00 WIB, dua hari kerja berikutnya setelah praktikum, kecuali ada kesepakatan lain antara Dosen Pengajar dan/atau Asisten.

Pergantian Jadwal

Kasus Biasa

Pergantian jadwal dilakukan dengan proses pertukaran. Pertukaran jadwal hanya dapat dilakukan per orang dengan modul yang sama. Langkah untuk menukar jadwal adalah sebagai berikut:

1) Lihatlah format Pertukaran Jadwal di http://ldte.stei.itb.ac.id pada halaman Panduan
2) Salah satu praktikan yang bertukar jadwal harus mengirimkan e-mail ke labdasar@stei.itb.ac.id atau melalui akun Official Line : @kiy3574q. Waktu pengiriman paling lambat jam 16.30, satu hari kerja sebelum praktikum yang dipertukarkan.
3) Pertukaran diperbolehkan setelah ada konfirmasi dari Lab. Dasar.

Kasus Sakit atau Urusan Mendesak Pribadi Lainnya

Jadwal pengganti dapat diberikan kepada praktikan yang sakit atau memiliki urusan mendesak pribadi. Praktikan yang hendak mengubah jadwal untuk urusan pribadi mendesak harus memberitahu staf tata usaha laboratorium sebelum jadwal praktikumnya melalui email.

Segera setelah praktikan memungkinkan mengikuti kegiatan akademik, praktikan dapat mengikuti praktikum pengganti setelah mendapatkan konfirmasi dari staf tata usaha laboratorium dengan melampirkan surat keterangan dokter bagi yang sakit atau surat terkait untuk yang memiliki urusan pribadi.

Kasus ”kepentingan massal”

”Kepentingan massal” terjadi jika ada lebih dari sepertiga rombongan praktikan yang tidak dapat melaksanakan praktikum pada satu hari yang sama karena alasan yang terkait kegiatan akademis, misalnya Ujian Tengah Semester pada jadwal kelompoknya. Beritahukan kepada administrasi TU Lab. Dasar secepatnya. Jadwal praktikum pengganti satu hari itu akan ditentukan kemudian oleh admin Lab. Dasar.

Sanksi

Pengabaian aturan-aturan di atas dapat dikenakan sanksi pengguguran nilai praktikum terkait.

Panduan Umum Keselamatan dan Penggunaan Peralatan

Keselamatan

Pada prinsipnya, untuk mewujudkan praktikum yang aman diperlukan partisipasi seluruh praktikan dan asisten pada praktikum yang bersangkutan. Dengan demikian, kepatuhan setiap praktikan terhadap uraian panduan pada bagian ini akan sangat membantu mewujudkan praktikum yang aman.

Bahaya Listrik

Perhatikan dan pelajari tempat-tempat sumber listrik (stop-kontak dan circuit breaker) dan cara menyala-matikannya. Jika melihat ada kerusakan yang berpotensi menimbulkan bahaya, laporkan pada asisten.

- Hindari daerah atau benda yang berpotensi menimbulkan bahaya listrik (sengatan listrik/ strum) secara tidak disengaja, misalnya kabel jala-jala yang terkelupas dll.
- Tidak melakukan sesuatu yang dapat menimbulkan bahaya listrik pada diri sendiri atau orang lain.
- Keringkan bagian tubuh yang basah karena, misalnya, keringat atau sisa air wudhu.
- Selalu waspada terhadap bahaya listrik pada setiap aktivitas praktikum.

Kecelakaan akibat bahaya listrik yang sering terjadi adalah tersengat arus listrik. Berikut ini adalah hal-hal yang harus diikuti praktikan jika hal itu terjadi:

- Jangan panik,
- Matikan semua peralatan elektronik dan sumber listrik di meja masing-masing dan di meja praktikan yang tersengat arus listrik,
- Bantu praktikan yang tersengat arus listrik untuk melepaskan diri dari sumber listrik,
- Beritahukan dan minta bantuan asisten, praktikan lain dan orang di sekitar anda tentang terjadinya kecelakaan akibat bahaya listrik.

Bahaya Api atau Panas berlebih

Jangan membawa benda-benda mudah terbakar (korek api, gas dll.) ke dalam ruang praktikum bila tidak disyaratkan dalam modul praktikum.

- Jangan melakukan sesuatu yang dapat menimbulkan api, percikan api atau panas yang berlebihan.
- Jangan melakukan sesuatu yang dapat menimbulkan bahaya api atau panas berlebih pada diri sendiri atau orang lain.

- Selalu waspada terhadap bahaya api atau panas berlebih pada setiap aktivitas praktikum.

Berikut ini adalah hal-hal yang harus diikuti praktikan jika menghadapi bahaya api atau panas berlebih:

- Jangan panik,
- Beritahukan dan minta bantuan asisten, praktikan lain dan orang di sekitar anda tentang terjadinya bahaya api atau panas berlebih,
- Matikan semua peralatan elektronik dan sumber listrik di meja masing-masing,
- Menjauh dari ruang praktikum.

Bahaya Lain

Untuk menghindari terjadinya hal-hal yang tidak diinginkan selama pelaksanaan percobaan perhatikan juga hal-hal berikut:

- Jangan membawa benda tajam (pisau, gunting dan sejenisnya) ke ruang praktikum bila tidak diperlukan untuk pelaksanaan percobaan.
- Jangan memakai perhiasan dari logam misalnya cincin, kalung, gelang dll.
- Hindari daerah, benda atau logam yang memiliki bagian tajam dan dapat melukai
- Hindari melakukan sesuatu yang dapat menimbulkan luka pada diri sendiri atau orang lain, misalnya bermain-main saat praktikum

Lain-lain

Praktikan dilarang membawa makanan dan minuman ke dalam ruang praktikum.

Penggunaan Peralatan Praktikum

Berikut ini adalah panduan yang harus dipatuhi ketika menggunakan alat-alat praktikum:

- Sebelum menggunakan alat-alat praktikum, pahami petunjuk penggunaan alat itu. Petunjuk penggunaan beberapa alat dapat didownload di http://ldte.stei.itb.ac.id.
- Perhatikan dan patuhi peringatan (warning) yang biasa tertera pada badan alat.
- Pahami fungsi atau peruntukan alat-alat praktikum dan gunakanlah alat-alat tersebut hanya untuk aktivitas yang sesuai fungsi atau peruntukannya. Menggunakan alat praktikum di luar fungsi atau peruntukannya dapat menimbulkan kerusakan pada alat tersebut dan bahaya keselamatan praktikan.
- Pahami rating dan jangkauan kerja alat-alat praktikum dan gunakanlah alat-alat tersebut sesuai rating dan jangkauan kerjanya. Menggunakan alat praktikum di luar rating dan jangkauan kerjanya dapat menimbulkan kerusakan pada alat tersebut dan bahaya keselamatan praktikan.

- Pastikan seluruh peralatan praktikum yang digunakan aman dari benda/ logam tajam, api/ panas berlebih atau lainnya yang dapat mengakibatkan kerusakan pada alat tersebut.
- Tidak melakukan aktifitas yang dapat menyebabkan kotor, coretan, goresan atau sejenisnya pada badan alat-alat praktikum yang digunakan.
- Kerusakan instrumentasi praktikum menjadi tanggung jawab bersama rombongan praktikum ybs. Alat yang rusak harus diganti oleh rombongan tersebut.

Sanksi

Pengabaian uraian panduan di atas dapat dikenakan sanksi tidak lulus mata kuliah praktikum yang bersangkutan

Tabel Sanksi Praktikum
Lab Dasar Teknik Elektro

Level	Waktu	Kasus	Sanksi	Pengurangan nilai per modul
Akademik	Saat dan setelah praktikum	Semua kegiatan plagiasi (mencontek): tugas pendahuluan, test dalam praktikum, laporan praktikum	Gugur praktikum	
		Sengaja tidak mengikuti praktikum		
Berat	Saat praktikum	Tidak hadir praktikum	Gugur modul	
		Terlambat hadir praktikum		
		Pakaian tidak sesuai: kemeja, sepatu		
		Tugas pendahuluan tidak dikerjakan/hilang/tertinggal		
Ringan	Saat Praktikum	Pertukaran jadwal tidak sesuai aturan/ketentuan		-25 nilai akhir
		Tidak mempelajari modul sebelum praktikum/tidak mengerti isi modul	Dikeluarkan dari praktikum	-25 nilai akhir
		BCL tertinggal/hilang		-100% nilai BCL
		Name Tag tertinggal/hilang		-10 nilai akhir
		Kartu praktikum tertinggal/hilang		-25 nilai akhir
		Kartu praktikum tidak lengkap data dan foto		-10 nilai akhir
		Loker tidak dikunci/kunci tertinggal		-10 nilai akhir
	Setelah Praktikum	Tidak ada paraf asisten di BCL/kartu praktikum		-25 nilai akhir
		Terlambat mengumpulkan laporan		-1/min nilai akhir, maks -50
		Terlambat mengumpulkan BCL		-1/min nilai BCL, maks -50
		Tidak bawa kartu praktikum saat pengumpulan BCL		-50 nilai BCL
		Tidak minta paraf admin saat pengumpulan BCL		-50 nilai BCL

Catatan:

1. *Pelanggaran akademik menyebabkan gugur praktikum, nilai praktikum E*
2. *Dalam satu praktikum, praktikan maksimal boleh melakukan*
   a. *1 pelanggaran berat dan 1 pelanggaran ringan; atau*
   b. *3 pelanggaran ringan*
3. *Jika jumlah pelanggaran melewati point 2, praktikan dianggap gugur praktikum.*
4. *Praktikan yang terkena sanksi gugur modul wajib mengganti praktikum pada hari lain dengan nilai modul tetap 0. Waktu pengganti praktikum ditetapkan bersama asisten. Jika praktikan tidak mengikuti ketentuan praktikum (pengganti) dengan baik, akan dikenakan sanksi gugur praktikum.*
5. *Setiap pelanggaran berat dan ringan dicatat/diberikan tanda di kartu praktikum*
6. *Waktu acuan adalah waktu sinkron dengan NIST*
7. *Sanksi yang tercantum di tabel adalah sanksi minimum.*
8. *Sanksi yang belum tercantum akan ditentukan kemudian.*

PLAGIARISME DAN KECURANGAN AKADEMIK

Plagiarisme merupakan salah satu bentuk kecurangan akademik. Definisi **plagiarisme** sesuai Peraturan Akademik ITB adalah **menggunakan kata-kata atau karya orang lain sebagai kata-kata atau karya sendiri dalam suatu kegiatan akademik tanpa menyebutkan acuan yang dipakai**. Plagiarisme bisa dilakukan secara sengaja, akibat kecerobohan, maupun tidak sengaja. Plagiarisme merupakan pelanggaran integritas akademik. **Prinsip kejujuran intelektual** menyiratkan bahwa semua anggota komunitas akademik harus mengakui peran pemilik gagasan awal dalam hal kata-kata dan data yang membentuk dasar untuk pekerjaan mereka sendiri. Mengakui karya orang lain sebagai milik anda memberi makna bahwa anda telah gagal menyelesaikan proses pembelajaran. Plagiarisme adalah sangat tidak etis dan memiliki konsekuensi serius bagi karir masa depan Anda sekaligus merusak reputasi institusi.

Bentuk-bentuk plagiarisme:

1. Mengutip kata demi kata (Verbatim)
2. Parafrase: menuliskan kembali karya hasil orang lain dengan mengubah kata atau mengubah urutan kalimat, dengan mengikuti struktur argumen orang lain tersebut tanpa menyebutkan acuan.
3. Kolusi: kolaborasi tidak sah antar mahasiswa tanpa atribusi terhadap bantuan dari luar yang diterima, atau tidak mengikuti sebenarnya pada peraturan kerja berkelompok
4. Kutipan tidak akurat: salah kutip atau mencantumkan referensi yang tidak pernah dikutip.
5. Apresiasi (*acknowledgement*) tidak akurat: tidak menyebutkan kontribusi pihak yang berkontribusi atau sebaliknya memberi apresiasi pada pihak yang tidak berkontribusi.
6. Menggunakan jasa pihak ketiga, profesional maupun tidak.

Prinsip menghindari plagiarisme:

1. Semua karya ilmiah harus dilandasi latar belakang, motivasi, dan lain sebagainya yang bisa dipertanggungjawabkan secara ilmiah. Adalah wajib untuk menggunakan referensi untuk mendukung ide-ide yang telah Anda kembangkan.
2. Dalam karya ilmiah, Anda harus menunjukkan bahwa Anda memiliki pemahaman yang jelas dan benar tentang materi yang telah Anda dapatkan dari referensi.
3. Berikan kejelasan antara analisa (ide) original Anda dengan apa yang telah diambil dari referensi:
    - Berikan penanda bagian mana suatu paragraf adalah berasal dari referensi.
    - Kutipan harus selalu diidentifikasi dengan menggunakan tanda kutip atau indentasi, dan dengan referensi penuh dari sumber yang dikutip.
    - Untuk menghindari parafrase, lebih baik menuliskan kembali ringkasan singkat dari keseluruhan sumber dengan kata-kata sendiri, dan dengan jelas menunjukkan bahwa itu yang dilakukan sehingga jelas bagian mana yang merupakan ide original Anda, mana yang diambil dari referensi.
    - Untuk menghindari kolusi, adalah tanggung jawab Anda untuk memastikan bahwa Anda sepenuhnya jelas tentang sejauh mana kolaborasi/kerja kelompok diizinkan, dan bagian mana dari pekerjaan itu harus Anda kerjakan sendiri.
    - Tidak boleh memasukkan apa pun dalam referensi atau bibliografi yang sebenarnya tidak direferensikan.

- Jika akses ke sumber utama tidak diperoleh, boleh menggunakan teks sekunder.
- Sitasi (menyebutkan) referensi harus diikuti dengan identifikasi pengutipannya dalam paragraf.

Kecurangan akademik dalam pelaksanaan praktikum

Tugas pendahuluan harus dikerjakan sendiri dalam setiap aspeknya, baik apabila tugas berupa analisis, perhitungan, atau simulasi. Kegiatan mencontoh atau meniru tugas pendahuluan tidak diperkenankan, dan apabila terbukti/bisa dibuktikan dapat dianggap melalukan kecurangan akademik seperti halnya mencontek. Apabila tugas yang diberikan membutuhkan referensi dari buku, internet dan sejenisanya, berlaku aturan plagiarisme. Untuk menghindari plagiarisme dalam mengerjakan tugas pendahuluan yang membutuhkan referensi, gunakan minimal 3 referensi dengan melakukan elaborasi dari referensi-referensi tersebut. Hindari dalam menggunakan hanya satu referensi meskipun dengan melakukan parafrase.

Tes awal termasuk dalam kategori yang sama dengan kuis atau **ujian**, dimana segala bentuk upaya mendapatkan bantuan dari pihak luar (mencontek pekerjaan peserta lain dengan bekerjasama atau tidak, menerima bantuan melalui alat komunikasi, memakai joki, dsb) dan menggunakan metode diluar yang diperkenankan (memakai contekan: melalui catatan, smartphone, dsb) adalah terlarang dan merupakan pelanggaran akademik.

Laporan praktikum sebagaimana laporan teknis, makalah, dan buku TA termasuk dalam kategori karya ilmiah, sehingga definisi dan aturan mengenai plagiarisme berlaku. Kecurangan yang biasa dilakukan diantaranya menggunakan data dari peserta lain, menggunakan template laporan peserta lain dan hanya mengganti datanya dan melakukan parafrase isi laporan yang lain.

MODUL 1

ATMega 328

1.1 TUJUAN

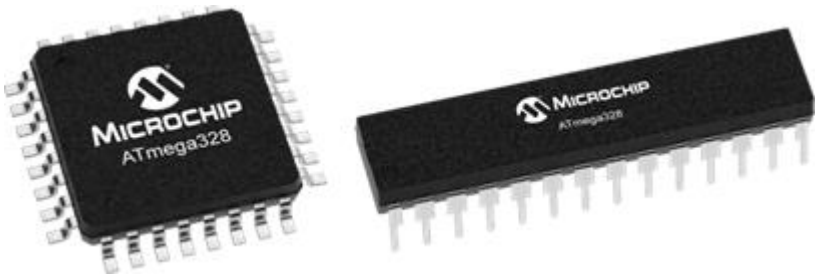
- Praktikan memahami datasheet ATMega 328
- Praktikan mampu membuat aplikasi input dan output pada AVR dengan menggunakan bahasa pemrograman C pada CodeVision dan Arduino IDE
- Praktikan mampu membuat aplikasi timer dan interrupt pada AVR dengan menggunakan bahasa pemrograman C pada CodeVision dan Arduino IDE

1.2 LANDASAN TEORI

1.2.1 ATMega328

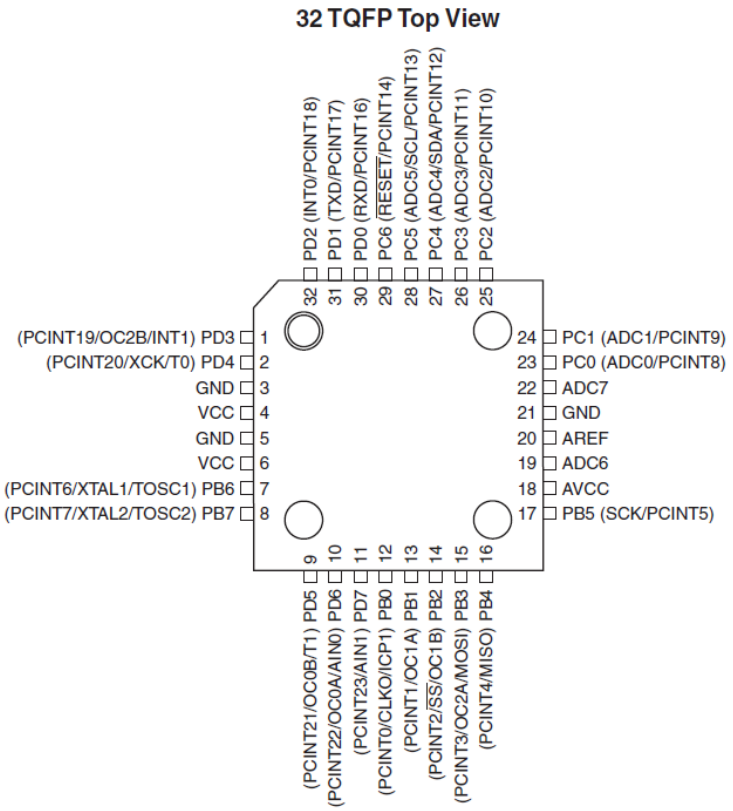
ATMega328 adalah mikrokontroler CMOS 8-bit daya rendah berbasis arsitektur RISC keluaran Microchip (sebelumnya Atmel). ATMega328 bersifat *single-cycle*, yaitu instruksi dieksekusi pada satu siklus *clock*. Sebuah ATMega328 dapat mencapai 1 MIPS (*million instructions per second*) per MHz, sehingga pada perancangan sistem dengan ATMega328 dapat dilakukan optimalisasi dalam konsumsi daya dengan tetap memperhatikan kecepatan pemrosesan.

ATMega328 diproduksi dalam empat jenis *package*, yaitu *thin quad flat pack* (TQFP), *shrink plastic dual in-line package* (SPDIP), *very thin quad-flat no-leads* (VQFN) 28 pin, dan VQFN 32 pin. Gambar berikut menunjukkan ATMega328 TQFP (kiri) dan SPDIP (kanan).

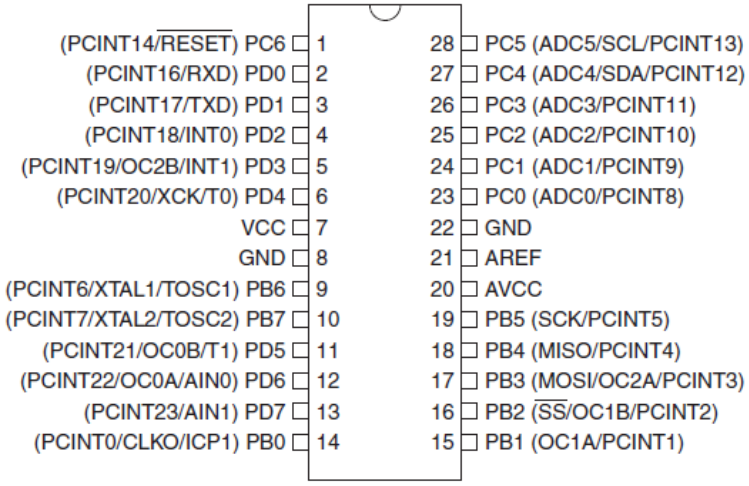


Gambar 1. 1 ATMega328

Konfigurasi pin untuk kedua jenis *package* ATMega tersebut dapat dilihat pada gambar berikut:



Gambar 1. 2 Pin ATMega328 TQFP

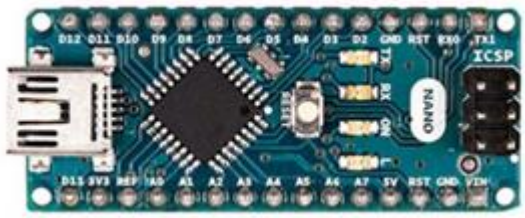


Gambar 1. 3 Pin ATMega328 SPDIP

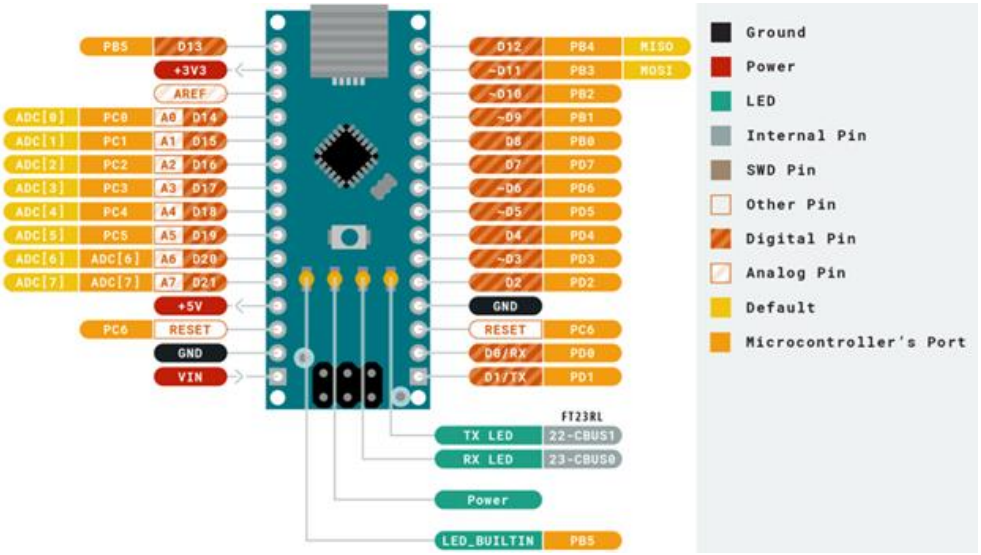
1.2.2 Arduino Nano

Arduino adalah platform pengembangan prototipe elektronika *open-source* berbasis perangkat keras dan perangkat lunak yang mudah digunakan. Arduino Nano adalah salah satu *board* pengembangan Arduino berbasis mikrokontroler

ATMega328. Dengan menggunakan *board* berbasis ATMega328 seperti Arduino Nano, proses perancangan prototipe menjadi lebih mudah dibanding langsung menggunakan ATMega328. Arduino Nano juga sudah dilengkapi dengan komponen-komponen tambahan seperti *pin header*/slot *pin header*, *voltage regulator*, LED, *external clock crystal*, *reset switch*, dll. Pada modul ini, percobaan terhadap ATMega328 dilakukan menggunakan Arduino Nano. Gambar-gambar berikut menunjukkan Arduino Nano beserta skematik *pinout*-nya.



Gambar 1. 4 Arduino Nano



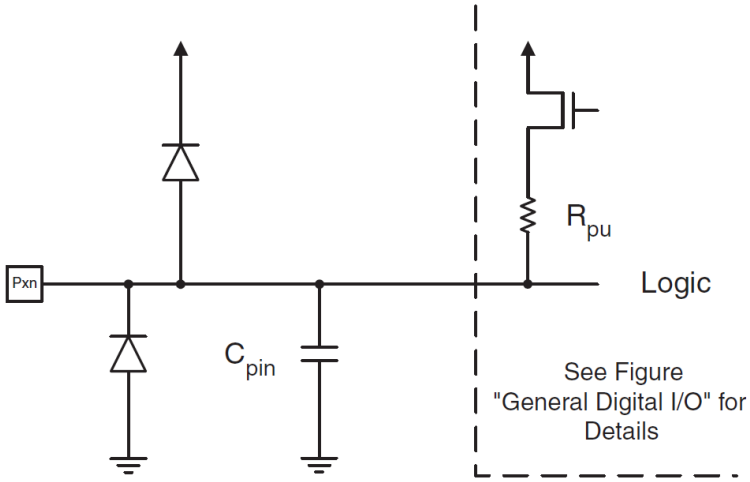
Gambar 1. 5 Pin Arduino Nano

Dikutip dari laman Arduino Store, Arduino Nano memiliki spek sebagai berikut:

Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog IN Pins	8
EEPROM	1 KB
DC Current per I/O Pins	40 mA (I/O Pins)
Input Voltage	7-12 V
Digital I/O Pins	22 (6 of which are PWM)
PWM Output	6
Power Consumption	19 mA
PCB Size	18 x 45 mm
Weight	7 g
Product Code	A000005

Gambar 1. 6 Spesifikasi Arduino Nano

1.2.3 Skematik I/O (Datasheet ATMega328 Hal. 84)



Gambar 1. 7 Skematik I/O

Huruf 'x' pada 'Pxn' di gambar skematik I/O tersebut merepresentasikan huruf penomoran *port*, sedangkan huruf 'n' merepresentasikan nomor bit *port*. Sebagai contoh, pin PB1 (pin paling kiri bawah pada skematik ATMega328 SPDIP) berarti *port* B bit 1.

1.2.4 Deskripsi Register dan Konfigurasi Pin Port

Masing-masing bit pada masing-masing *port* dapat dikonfigurasi pada register PORTx, DDRx, dan PINx. Pengaturan nilai bit PORTxn, DDRxn, dan PINxn akan berefek pada Pxn. Sebagai contoh, berikut adalah register untuk *port* B.

PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PINB – The Port B Input Pins Address(1)

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Gambar 1. 8 Register I/O

Gambar berikut menunjukkan nilai bit serta konfigurasinya:

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

Gambar 1. 9 Konfigurasi I/O Arduino Nano

1.2.5 PUD pada MCUCR (Datasheet ATMega328 Hal. 100)

MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	BODS(1)	BODSE(1)	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Notes: 1. BODS and BODSE only available for picoPower devices ATmega48PA/88PA/168PA/328P

Gambar 1. 10 MCUCR

Ketika bit 4 – PUD (*pull-up disable*) pada MCUCR tersebut bernilai 1, *pull-up* pada *port* I/O akan dinonaktifkan walaupun pada register DDxn dan PORTxn *pull-up* diset aktif.

1.2.6 Rekomendasi untuk Inisialisasi pada Pin Input

Pin yang tidak digunakan disarankan untuk diatur agar tetap memiliki level tegangan yang terdefinisi. Walaupun sebagian besar input digital tidak aktif pada mode *deep sleep*, input *floating* harus dihindari untuk mengurangi konsumsi arus pada mode lainnya ketika input digital diaktifkan (saat reset, mode aktif, dan mode *idle*).

Cara paling sederhana untuk memastikan bahwa setiap pin yang tidak terpakai memiliki level tegangan yang terdefinisi adalah dengan mengaktifkan *internal pull-up*. Hal tersebut akan menyebabkan *pull-up* menjadi tidak aktif saat reset. Jika diinginkan agar konsumsi daya rendah pada saat reset, disarankan untuk menggunakan *pull-up* atau *pull-down* eksternal. Tidak disarankan untuk menghubungkan pin yang tidak digunakan ke VCC atau GND secara langsung karena dapat menyebabkan aliran arus berlebih jika pin secara tidak sengaja dikonfigurasi sebagai output.

1.2.7 Interrupt (Datasheet ATMega328 Hal. 20)

Pada dasarnya, program yang berjalan pada mikrokontroler akan dieksekusi secara sekuensial. Akan tetapi, eksekusi program tersebut dapat diinterupsi ketika suatu kondisi tercapai. Program akan kembali dilanjutkan jika instruksi-instruksi pada *interrupt service routine* (ISR) selesai dieksekusi.

Pada ATMega328, agar *interrupt* dapat digunakan, bit SREG I atau *Global Interrupt Enable* harus aktif. Kemudian pengaktifan masing-masing *interrupt* dilakukan di register kontrol terpisah. Jika register *Global Interrput Enable* tidak aktif, tidak ada *interrupt* yang dapat aktif.

SREG – AVR Status Register

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 1. 11 Status Register

1.2.8 External Interrupt (Datasheet ATMega328 Hal. 80 s/d 81)

Deskripsi register kontrol *external interrupt*:

EICRA – External Interrupt Control Register A

The External Interrupt Control Register A contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 1. 12 EICRA

Bit 7:2 – *Reserved*: Bit ini tidak digunakan pada ATMega328 dan akan selalu bernilai 0.

Bit 3, 2 – ISC11, ISC10 (*Interrupt Sense Control* 1 Bit 1 dan Bit 0): Serupa dengan ISC0n, tetapi dipicu oleh pin eksternal INT1.

Bit 1, 1 – ISC01, ISC00 (*Interrupt Sense Control* 0 Bit 1 dan Bit 0): *External Interrupt* 0 diaktifkan oleh pin eksternal INT0 jika bit SREG I dan *interrupt mask* yang sesuai aktif. Nilai pin INT1 disampel terlebih dahulu sebelum mendeteksi *edge* (*rising* atau *falling*). Jika dipilih *edge interrupt*, pulsa dengan durasi lebih dari 1 perioda *clock* akan mengakibatkan *interrupt*. Pulsa dengan durasi lebih singkat tidak dipastikan untuk mengakibatkan *interrupt*. Jika *low level interrupt* dipilih, *low level* harus dipertahankan sampai instruksi yang sedang dieksekusi pada saat itu selesai agar *interrupt* terjadi. Level dan *edge* yang dapat mengaktifkan ISCxn didefinisikan pada gambar berikut.

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

Gambar 1. 13 Definisi ISCxn

Deskripsi register *interrupt mask* untuk *external interrupt*:

EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 1. 14 EIMSK

Bit 7:2 – *Reserved*: Bit ini tidak digunakan pada ATMega328 dan akan selalu bernilai 0.
Bit 1 – INT1: *External Interrupt Request* 1 *Enable*
Bit 0 – INT0: *External Interrupt Request* 0 *Enable*

Deskripsi register untuk *flag* dari *external interrupt*:

EIFR – External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	–	–	–	–	INTF1	INTF0	EIFR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 1. 15 EIFR

Bit 7:2 – *Reserved*: Bit ini tidak digunakan pada ATMega328 dan akan selalu bernilai 0.
Bit 1 – INTF1: *External Interrupt Flag* 1
Bit 0 – INTF0: *External Intterupt Flag* 0

1.2.9 Timer Interrupt pada ATMega328

Selain dari pin eksternal, *interrupt* juga dapat dilakukan menggunakan *timer*, sehingga *interrupt* terjadi setelah durasi waktu tertentu. Pada ATMega328, *timer interrupt* dapat terjadi ketika *timer overflow flag* (TOV) aktif atau ketik *output compare match flag* (OCF) aktif. TOV aktif ketika nilai *timer* pada *Timer/Counter* (TCNT) mencapai *overflow*, atau melebihi batas nilai 8-bit,

sedangkan TOV aktif ketika nilai TCNT sama dengan nilai pada output compare register 0 (OCR0x).

Untuk menggunakan mode *timer overflow* agar interrupt terjadi setelah t detik, dilakukan inisialisasi suatu nilai pada TCNT, yaitu:

$$TCNT = 2^{bittimer} - \frac{frekuensi\ clock}{prescaler} \cdot t$$

Dengan demikian, waktu yang dibutuhkan agar TCNT mencapai *overflow* dari nilai awal TCNT adalah sebesar t detik.

Untuk menggunakan mode *output compare match* agar interrupt terjadi setelah t detik, diberikan nilai pada OCR0x, yaitu:

$$OCRx = \frac{frekuensi\ clock}{prescaler} \cdot t$$

Dengan demikian, waktu yang dibutuhkan agar TCNT memiliki nilai yang sama dengan OCR0x dari 0 adalah sebesar t detik.

1.2.10 Register-Register Penting untuk Timer Interrupt pada ATMega328

Deskripsi register TCCR0A – *Timer/Counter Control Register* A:

TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 1. 16 TCCR0A

Deskripsi bit COM0An untuk mode *non-PWM* (berlaku juga untuk COM0Bn):

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match
1	1	Set OC0A on Compare Match

Gambar 1. 17 Deskripsi bit COM0An

Deskripsi bit WGM01 dan WGM00:

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Gambar 1. 18 Deskripsi bit WGM01 dan WGM00

Deskripsi register TCNT0 – *Timer/Counter Register*:

TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 1. 19 Deskripsi register TCNT0

Register *Timer/Counter* memberikan akses langsung untuk operasi *read* dan *write* pada 8-bit *counter Timer/Counter*. Proses *write* pada register TCNT0 akan memblokir *Compare Match* sehingga modifikasi pada *counter* TCNT0 ketika *counter* sedang berjalan dapat menimbulkan resiko terlewatnya *Compare Match* antara register TCNT0 dan OCR0x.

Deskripsi register OCR0A – *Output Compare Register* A (berlaku juga untuk OCR0B):

OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x27 (0x47)	OCR0A[7:0]								OCR0A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 1. 20 Deskripsi register OCR0A

Output Compare Register A memiliki sebuah nilai 8-bit yang secara terus menerus dibandingkan dengan nilai *counter* (TCNT0). Ketika OCR0A dan *counter* bernilai sama, dihasilkan *Output Compare interrupt* atau gelombang sinyal output ke pin OC0A.

Deskripsi register TIMSK0 – *Timer/Counter Interrupt Mask Register*:

TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 1. 21 Deskripsi register TIMSK0

Bit 7:3 – *Reserved*: Bit ini tidak digunakan pada ATMega328 dan akan selalu bernilai 0.

Bit 2 – OCIE0B: *Timer/Counter* 0 *Output Compare Match* B *Interrupt Enable*

Bit 1 – OCIE0B: *Timer/Counter* 0 *Output Compare Match* A *Interrupt Enable*

Bit 0 – TOIE0: *Timer/Counter* 0 *Overflow Interrupt Enable*

Deskripsi register TIFR0 – *Timer/Counter* 0 *Interrupt Flag Register*:

TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 1. 22 Deskripsi register TIFR0

Bit 7:3 – *Reserved*: Bit ini tidak digunakan pada ATMega328 dan akan selalu bernilai 0.

Bit 2 – OCF0B: *Timer/Counter* 0 *Output Compare* B *Match Flag*

Bit 1 – OCF0A *Timer/Counter* 0 *Output Compare* A *Match Flag*

Bit 0 – TOV0: *Timer/Counter* 0 *Overflow Flag*

1.2.11 Pengaturan Sumber Clock (Datasheet ATMega328 Hal. 116 s/d 117)

Deskripsi register TCCR0B – *Timer/Counter Control Register* B:

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 1. 23 Deskripsi register TCCR0B

Deskripsi bit CS02, CS01, dan CS00:

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Gambar 1. 24 Deskripsi bit CS02, CS01, dan CS00

Referensi

> *Datasheet* ATMega328
> https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf
>
> Arduino Nano
> https://store.arduino.cc/usa/arduino-nano
> https://www.arduino.cc/en/Main/Standalone
>
> *Timer Interrupt*
> https://www.robotshop.com/community/forum/t/arduino-101-timers-and-interrupts/13072

1.3 TUGAS PENDAHULUAN

1. Membuat semua program (source code) yang diperlukan untuk masing-masing percobaan (sertakan keterangan-keterangan penting pada source code menggunakan komentar); Jelaskan masing-masing baris atau bagian kode tersebut.

1.4 ALAT DAN KOMPONEN YANG DIGUNAKAN

- Arduino Nano (1 buah)
- LED (8 buah)
- Push Button (1 buah)
- Resistor 1k (9 buah)
- Kabel Jumper (Secukupnya)
- Laptop dengan Codevision dan Arduino IDE (1 buah)
- 7Segment (1 buah)

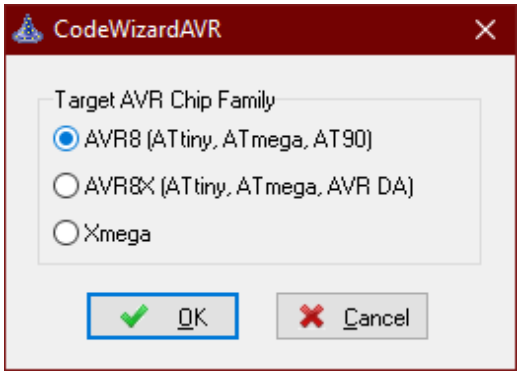
1.5 MENGGUNAKAN CODEVISION AVR

Persiapan / Setting awal

- Install CodeVision AVR eval

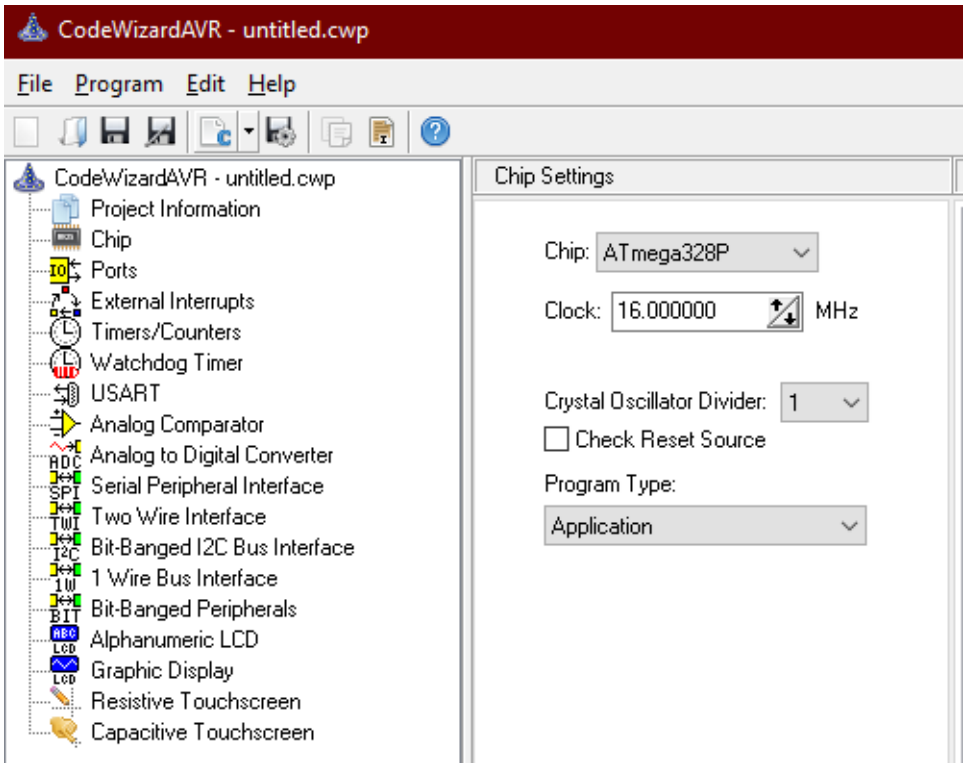
1.5.1 Output Digital

1. Jalankan *software* CodeVisionAVR. Buat *project* baru dengan klik **File > New > Project > Yes**. Pilih **AVR8** pada menu selanjutnya, kemudian klik **OK**.



Gambar 1. 25 Jendela Wizard 1

2. Pada *chip settings*, pilih ATmega328P untuk pilihan Chip.



Gambar 1. 26 Jendela Wizard 2

3. Kemudian klik **Program > Generate, Save and Exit**. Akan dihasilkan tiga file yang harus di *save*, *file* (.prj) C *file* (.c). Simpan kedua file dengan nama **tugas151**. Program akan menghasilkan file bahasa C secara otomatis sesuai konfigurasi yang telah diatur.

4. Ganti *source code* yang telah dihasilkan dengan *source code* berikut. Lengkapi bagian kode sesuai dengan komentar pada kode.

Tabel 1. 1 Kode Output Digital

```
#include <mega328.h>
#include <delay.h>
```

```
int i;

void main(void) {
  //Set mode PIN D menjadi OUTPUT
  DDRD=0bXXXXXXXX;

  while (1)  {
    /*=============================================
    Tuliskan Kode Program yang membuat perilaku running
    LED dari LED LSB ke MSB. Pergantian LED diberi jeda
    sebesar 1 detik. Ilustrasi output sebagai berikut
      X 0 0 0 0 0 0 0   X = ON
      0 X 0 0 0 0 0 0   0 = OFF
      0 0 X 0 0 0 0 0
      0 0 0 X 0 0 0 0
      0 0 0 0 X 0 0 0
      0 0 0 0 0 X 0 0
      0 0 0 0 0 0 X 0
      0 0 0 0 0 0 0 X
      X 0 0 0 0 0 0 0
      dan seterusnya...
    =============================================*/
  }
}
```

5. Selanjutnya klik **Project > Compile** untuk melakukan kompilasi kode. Bisa juga dilakukan dengan klik F9. Pastikan tidak terdapat *error*.

6. Kemudian klik **Project > Build All** atau klik Ctrl + F9. Software akan melakukan proses *build* dan menghasilkan file HEX. Pastikan kembali tidak terdapat *error*.

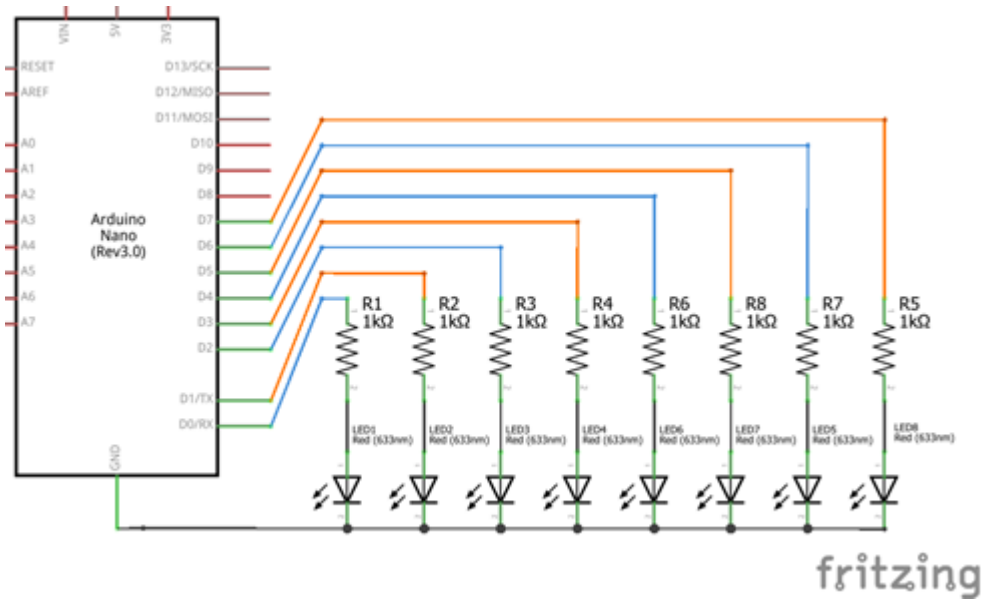
Selanjutnya akan dilakukan perangkaian komponen untuk menjalankan program.

7. Siapkan Arduino Nano, LED, Resistor 1kΩ serta kabel *jumper* yang dibutuhkan.

8. Lakukan hubungkan untuk resistor 1kΩ dengan anoda LED. Lakukan untuk kedelapan LED.

9. Hubungkan semua katoda LED dengan pin GND pada Arduino Nano.

10. Hubungkan ujung resistor yang belum terhubung dengan pin RX0, TX1, D2, D3, D4, D5, D6,dan D7 secara berurutan sehingga rangkaian sesuai dengan skematik berikut.

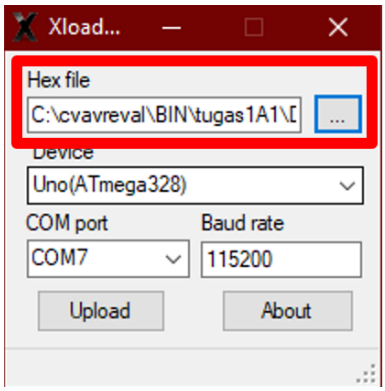


Gambar 1. 27 Skema Rangkaian Tugas 1.5.1

Selanjutnya akan dilakukan proses upload program ke Arduino Nano yang telah disiapkan.

11. Buka aplikasi **Xloader.exe** yang telah diinstall. Langkah instalasi dapat dilihat pada bagian Lampiran.

12. Pilih HEX file yang telah dihasilkan pada CodeVisionAVR. File .hex terdapat pada folder **[Nama File Project] > Debug > Exe > tugas151.hex**



Gambar 1. 28 Jendela Xloader

13. Pilih **Duemilanove/Nano(ATmega328)** pada menu *Device*.

14. Pilih COM Port sesuai dengan Port pada PC yang dihubungkan dengan Arduino Nano. Kemudian klik Upload.

15. Jika berhasil, Rangkaian Arduino Nano akan menghasilkan *output* sesuai deskripsi pada program

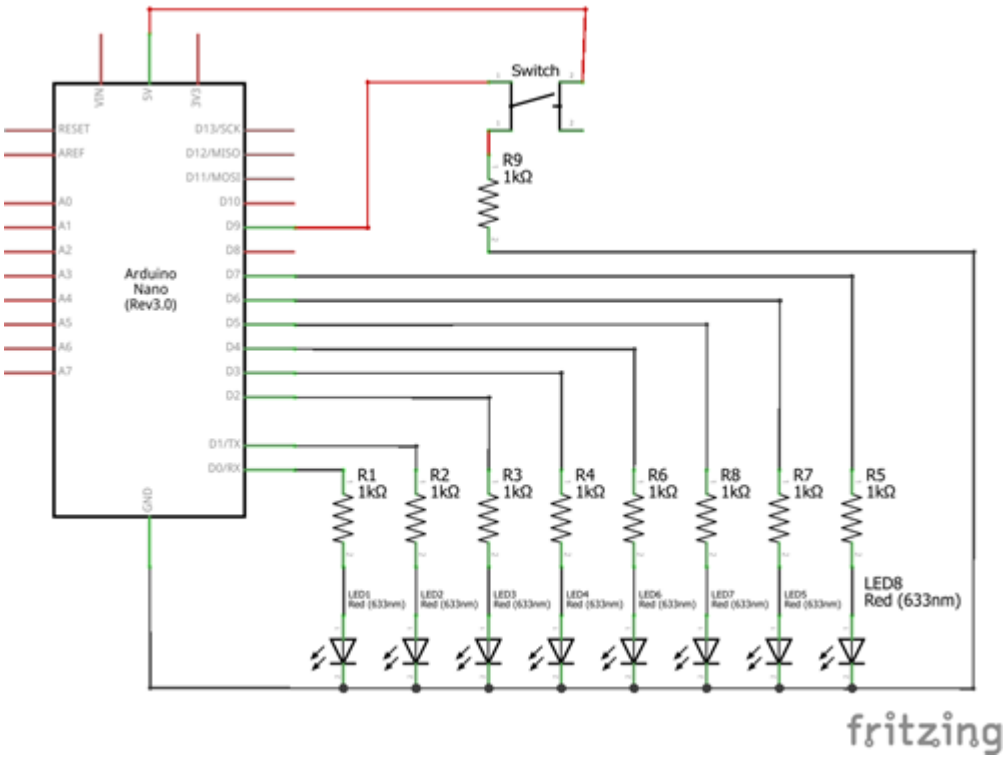
1.5.2 Input Digital

1. Buat *project* baru, simpan dengan nama **tugas152**.
2. Modifikasi *source code* menjadi seperti berikut. Lengkapi bagian kode sesuai dengan komentar pada kode.

Tabel 1. 2 Kode Input Digital

```
#include <mega328.h>
#include <delay.h>

void main(void) {
  //Set mode PIN D menjadi OUTPUT
  DDRD=0bXXXXXXXX;

  //Set mode PIN B menjadi INPUT
  DDRB=0bXXXXXXXX;

  while (1)  {
    /*================================================
    Tuliskan Kode Program yang membuat perilaku running
    LED menyala. Namun, perilaku LED yang menyala
    terjadi setiap push button ditekan. Ilustrasi output
    sebagai berikut.
      X 0 0 0 0 0 0 0          X = ON
      *push button ditekan*    0 = OFF
      X X 0 0 0 0 0 0
      *push button ditekan*
      X X X 0 0 0 0 0
      *push button ditekan*
      X X X X 0 0 0 0
      *push button ditekan*
      X X X X X 0 0 0
      ...
      X X X X X X X X
      *push button ditekan*
      0 0 0 0 0 0 0 X
      *push button ditekan*
      0 0 0 0 0 0 X X
      dan seterusnya...
    ================================================*/
  }
}
```

3. Lakukan *compile* dan *build*. Pastikan kembali tidak terdapat *error*.
4. Tambahkan *push button* dengan konfigurasi *pull-down* sehingga rangkaian sesuai dengan skematik berikut.



Gambar 1. 29 Skema Rangkaian Tugas 1.5.2

5. Upload file .hex ke *board* Arduino Nano dengan Xloader.exe.
6. Jika berhasil, Rangkaian Arduino Nano akan menghasilkan *output* sesuai deskripsi pada program.
7. Amati perilaku push button ketika ditekan dengan osiloskop.
8. Lakukan pengukuran tegangan pin I/O terhadap GND, LED terhadap GND, serta arus pada salah satu LED yang menyala.
9. Amati tegangan pada pin 5V terhadap setiap jumlah nyala LED. Misal, tegangan pin 5V pada saat 1 buah LED menyala sampai 8 buah LED menyala.

1.5.3 Timer

1. Unduh file .rar dari tautan berikut.

   https://drive.google.com/file/d/1V-smrjlxaYaRaBAYuXJRjKtmt6YEjj0h/view?usp=sharing

2. Tutup CodeVisionAVR. Kemudian letakkan hasil *extract* include.rar pada **C:\cvavreval\INC**
3. Buka kembali CodeVisionAVR. Kemudian buat *project* baru, simpan dengan nama **tugas153**.
4. Modifikasi *source code* menjadi seperti berikut. Lengkapi bagian kode sesuai dengan komentar pada kode.

Tabel 1. 3 Kode Timer

```
#include <mega328.h>
#include <avr/io.h>

void init_timer(void) {
  //mode operasi CTC, sumber clock prescaler 256
  TCCR1A = 0;
  TCCR1B = 0b0000XXXX;

  //overflow interrupt disable
  TIMSK1 |=(0<<TOIE1);

}

void Delay(void) {
  //Overflow interrupt enable
  TIMSK1 |= (1<<TOIE1);

  //Counter start value
  TCNT1H = 0;
  TCNT1L = 0;

  //Set OCR value for 1 Hz
  OCR1AH = 0xHH;
  OCR1AL = 0xHH;

  //Compare Match TCNT1 dan OCR1A
  loop_until_bit_is_set(TIFR1,OCF1A);

  //Reset Flag
  TIFR1 |= (1<<OCF1A);

  //overflow interrupt disable
  TIMSK1 |= (0<<TOIE1);
}

void main(void) {
  /*================================================
  Tuliskan Kode Program yang membuat output flip flop
  LED dengan pola 4 bit. Jeda antar flip flop sebesar
  500 ms. DILARANG MENGGUNAKAN FUNGSI delay_ms() unt-
  -uk menghasilkan jeda, gunakan fungsi Delay() untuk
  menghasilkan jeda. Ilustrasi output sebagai berikut
  0 0 0 0 X X X X          X = ON
  X X X X 0 0 0 0          0 = OFF
  0 0 0 0 X X X X
  X X X X 0 0 0 0
  dan seterusnya...
  ================================================*/
}
```

5. Lakukan *compile* dan *build*. Pastikan kembali tidak terdapat *error*.
6. Upload file .hex ke *board* Arduino Nano dengan Xloader.exe
7. Jika berhasil, Rangkaian Arduino Nano akan menghasilkan *output* sesuai deskripsi pada program.

1.5.4 Interupsi Timer

1. Buat *project* baru, simpan dengan nama **tugas154**.
2. Modifikasi *source code* menjadi seperti berikut. Lengkapi bagian kode sesuai dengan komentar pada kode.

Tabel 1. 4 Kode Interupsi Timer

```
#include <mega328.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <delay.h>

//SET COUNT VALUE FOR 1 HZ
#define TCNT_HIGH 0xHH
#define TCNT_LOW 0xHH

void init_int(void) {
  //set prescaler 1024
  TIMSK1=0b000000;
  TCCR1B=0b0000XXXX;

  TCNT1H=TCNT_HIGH;
  TCNT1L=TCNT_LOW;

  //Enable Interrupt
  #asm("sei")
}


interrupt [TIM1_OVF] void timer1_ovf_isr(void) {
  //Tuliskan kode perilaku interrupt disini
}


void main(void) {
  init_int();
  while (1) {
    /*================================================
    Tuliskan Kode Program yang output looping nyala LED
    dengan interupsi setiap 1 detik. Output looping LED
    memiliki pola berulang dengan jeda 100 ms dengan i-
    -lustrasi sebagai berikut.
      X 0 X 0 X 0 X 0   X = ON
      0 X 0 X 0 X 0 X   0 = OFF
```

```
      X 0 X 0 X 0 X 0
      0 X 0 X 0 X 0 X
      dan seterusnya...
    Kemudian akan terjadi interrupt yang terjadi setiap
    1 detik. Interrupt akan menghasilkan output  berupa
    semua LED menyala selama 1 detik.
      *interrupt terjadi*
      X X X X X X X X
      *1 detik berlalu*
      interrupt selesai
    ================================================*/
  }
}
```

3. Lakukan *compile* dan *build*. Pastikan kembali tidak terdapat *error*.
4. Upload file .hex ke *board* Arduino Nano dengan Xloader.exe
5. Jika berhasil, Rangkaian Arduino Nano akan menghasilkan *output* sesuai deskripsi pada program.

1.5.5 Interupsi Tombol

1. Buat *project* baru, simpan dengan nama **tugas155**.
2. Modifikasi *source code* menjadi seperti berikut. Lengkapi bagian kode sesuai dengan komentar pada kode.

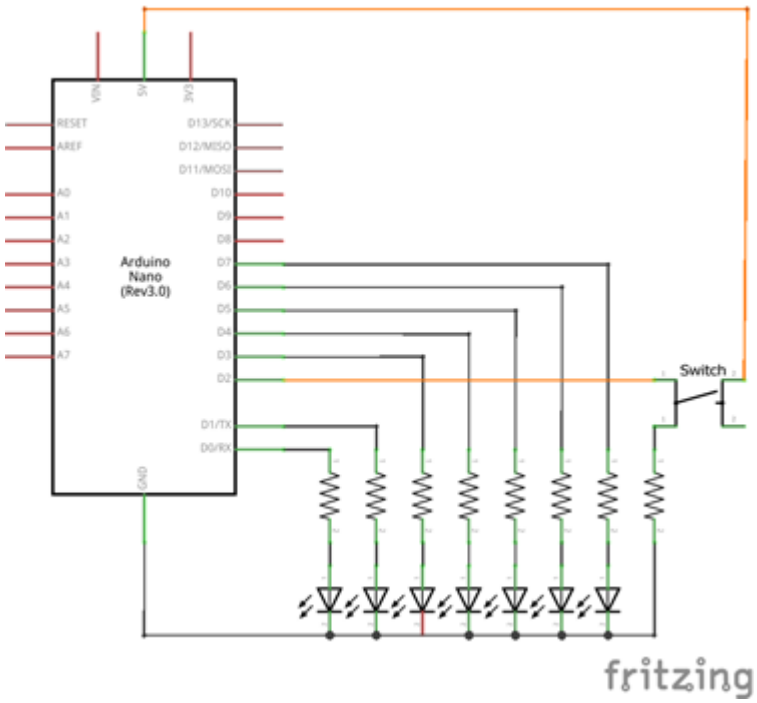
Tabel 1. 5 Kode Interupsi Tombol

```
#include <mega328.h>
#include <delay.h>

interrupt [EXT_INT0] void ext_int0_isr(void) {
  //Tuliskan kode perilaku interrupt disini
}

void main(void) {
  //SET MODE PIN D, PIN D2 = INPUT, SISANYA OUTPUT
  DDRD = 0xHH;
  #asm("sei")

  //SET FALLING EDGE PADA INT0
  EICRA=(X<<ISC11)|(X<<ISC10)|(X<<ISC01)|(X<<ISC00);

  //ENABLE INT0
  EIMSK=(X<<INT1) | (X<<INT0);

  //SET INT0 FLAG
  EIFR=(X<<INTF1) | (X<<INTF0);

  while (1) {
```

```
    /*================================================
    Tuliskan Kode Program yang output looping nyala LED
    dengan interupsi setiap push button ditekan. Output
    looping LED memiliki pola flip-flop yang sama deng-
    an tugas 1.5.3. Setiap push button ditekan, akan terj-
    adi interrupt dimana semua LED mati selama 250 ms.
    Berikut merupakan contoh perilaku.
      0 0 0 0 X X X X         X = ON
      X X X X 0 0 0 0         0 = OFF
      0 0 0 0 X X X X
      *interrupt push button*
      0 0 0 0 0 0 0 0
      *setelah 250 ms interrupt selesai*
      X X X X 0 0 0 0
      0 0 0 0 X X X X
      X X X X 0 0 0 0
      dan seterusnya...
    ================================================*/
  }
}
```

3. Lakukan *compile* dan *build*. Pastikan kembali tidak terdapat *error*.

4. Karena INT0 terdapat pada port D2, masukan *push button* harus dipindahkan dari port D9 ke port D2. Sehingga rangkaian sesuai dengan skematik berikut.



Gambar 1. 30 Skema Rangkaian Tugas 1.5.5

5. Upload file .hex ke *board* Arduino Nano dengan Xloader.exe

6. Jika berhasil, Rangkaian Arduino Nano akan menghasilkan *output* sesuai deskripsi pada program.

1.6 MENGGUNAKAN ARDUINO IDE

1.6.1 Output Digital

1. Jalankan aplikasi Arduino IDE. Buat *sketch* baru melalui **File > New** atau menekan Ctrl+N pada keyboard.
2. Buatlah *sketch* yang menghasilkan *output* seperti pada tugas 1.5.1.
3. Simpan *sketch* dengan nama **tugas161**.
4. Lakukan *compile* dengan menu **Sketch > Verify/Compile**. Proses *compile* juga bisa dilakukan dengan menekan Ctrl+R pada *keyboard* atau menekan tombol berikut.

Gambar 1. 31 Jendela Arduino IDE 1

5. Susun rangkaian sesuai dengan Skema Rangkaian Tugas 1.5.1.
6. Hubungkan Arduino Nano dengan PC menggunakan kabel USB.
7. Pilih Arduino Nano sebagai *board* pilihan pada **Tools > Board**
8. Pilih ATmega328P sebagai *processor* pilihan pada **Tools > Processor**
9. Pilih COM Port yang digunakan untuk menghubungkan Arduino Nano pada **Tools > Port**
10. Lakukan *upload* melalui menu **Sketch > Upload**. Proses *upload* juga bisa dilakukan dengan menekan Ctrl+U pada *keyboard* atau menekan tombol berikut.

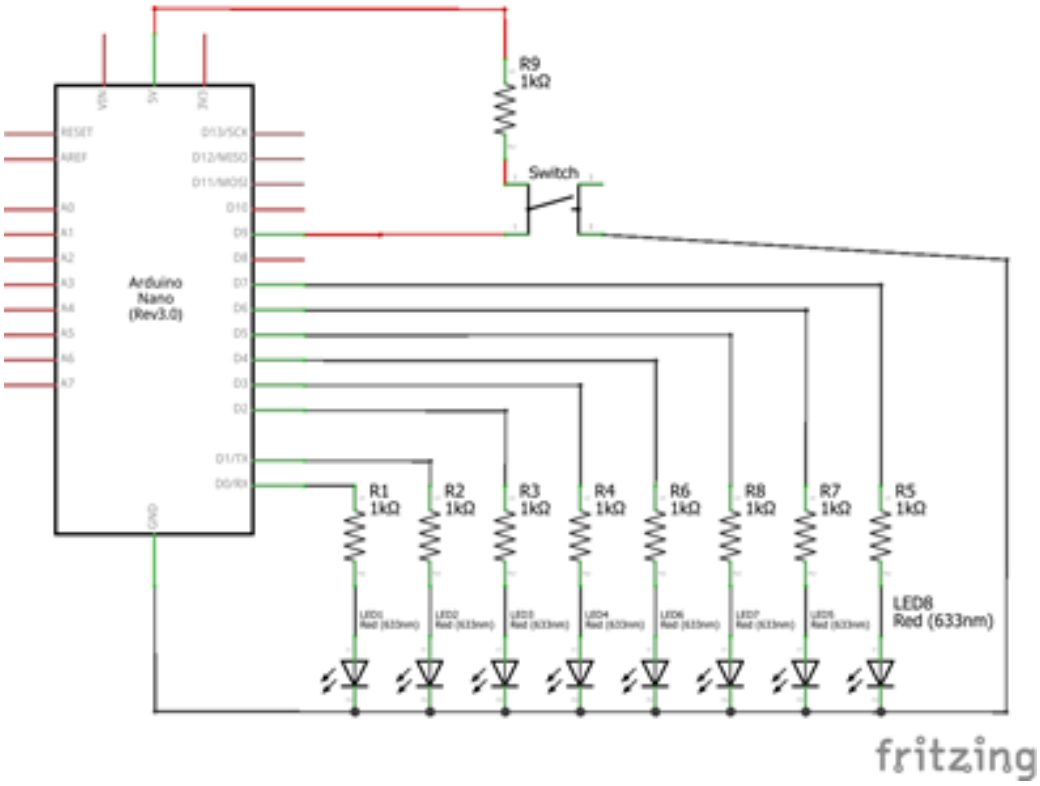
Gambar 1. 32 Jendela Arduino IDE 2

11. Pastikan *output* yang dihasilkan sesuai dengan perintah praktikum.

1.6.2 Input Digital

1. Buatlah *sketch* yang menghasilkan *output* yang sama seperti *output* pada Tugas 1.5.2.
2. Simpan *sketch* dengan nama file tugas162.

3. Susun rangkaian sesuai dengan Skema Rangkaian Tugas 1.5.2.
4. Lakukan proses *compile* kemudian *upload* program ke Arduino Nano.
5. Setelah berhasil dilakukan *upload*, pastikan *output* yang dihasilkan sesuai dengan perintah praktikum.
6. Modifikasi rangkaian button hingga menjadi *pull-up button* seperti skema rangkaian berikut.



Gambar 1. 33 Skema Rangkaian Tugas 1.6.2

Amati perbedaan yang terjadi kemudian modifikasi *sketch* sehingga *output* yang dihasilkan kembali seperti semula tanpa mengubah konfigurasi *push button*

1.6.3 Timer

1. Buatlah *sketch* yang menghasilkan *output* seperti pada tugas 1.5.3. **Untuk mengatur jeda gunakanlah *timer counter* pada Arduino Nano, tidak boleh menggunakan fungsi `delay()` yang telah tersedia**. Diperbolehkan menggunakan *library* TimerOne untuk mengatur *timer counter*.
2. Simpan *sketch* dengan nama *file* tugas163.
3. Lakukan proses *compile* kemudian *upload* program ke Arduino Nano.
4. Setelah berhasil dilakukan upload, pastikan *output* yang dihasilkan sesuai dengan perintah praktikum.

1.6.4 Interupsi Timer

1. Buatlah *sketch* yang menghasilkan *output* seperti pada tugas 1.5.4. L*ibrary* TimerOn*e* dapat digunakan untuk mengatur *timer interrupt*.
2. Simpan *sketch* dengan nama *file* tugas164.
3. Lakukan proses *compile* kemudian *upload* program ke Arduino Nano.
4. Setelah berhasil dilakukan upload, pastikan *output* yang dihasilkan sesuai dengan perintah praktikum seperti ilustrasi berikut.

1.6.5 Interupsi Tombol

1. Buatlah *sketch* yang menghasilkan *output* seperti pada tugas 1.5.5. *Gunakan* l*ibrary* ISR untuk mengatur *external interrupt.*
2. Simpan *sketch* dengan nama *file* tugas165.
3. Karena PIN D2 merupakan *input* untuk INT0, maka susun rangkaian sehingga *pull-up button* terhubung dengan PIN D2 pada Arduino Nano. Sehingga hanya digunakan 7 LED saja.
4. Lakukan proses *compile* kemudian *upload* program ke Arduino Nano.
5. Setelah berhasil dilakukan upload, pastikan *output* yang dihasilkan sesuai dengan perintah praktikum seperti ilustrasi berikut.

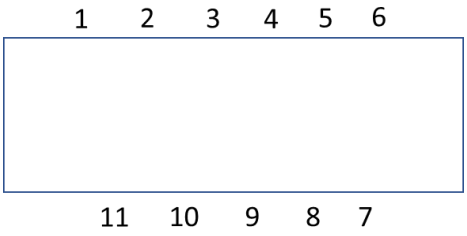
1.6.6 7 Segment

Buatlah sebuah program untuk mengimplementasikan sebuah counter sederhana yang menggunakan 7 segment 3 digit. Program melakukan counting setiap 1 detik, dari 0 hingga 999, dan ditampilkan pada 7segment. Counter diulang dari 0 ketika sudah mencapai 999. Program harus menggunakan interrupt timer.

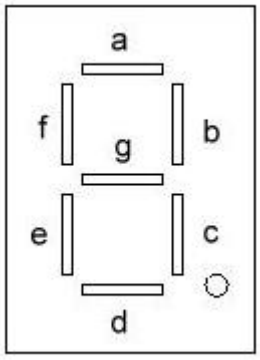
7 segment yang digunakan memiliki 11 pin. 3 pin untuk menentukan digit yang dipilih dan 8 pin untuk menyalakan 7 segment.

Perhatikan konfigurasi 7 segment, apakah common anode atau common cathode. Misal untuk konfigurasi common anode, jika diinginkan digit 1, maka pin 5 diset HIGH, sementara pin 6 dan 8 diset LOW. Pin 1 dan 4 diset LOW, serta pin sisanya diset HIGH.

Jangan lupa menggunakan resistor sebagai pembatas arus untuk masing-masing dioda. Perhatikan juga multiplexing antar digit, khususnya ketika transisi dari 1 angka ke angka lain.



Gambar 1. 34 Konfigurasi 7 Segment 1



Gambar 1. 35 Konfigurasi 7 Segment 2

Tabel 1. 6 Keterangan Konfigurasi 7 Segment

Pin	Keterangan
1	Digit 3 (kiri)
2	a
3	f
4	Digit 2 (tengah)
5	Digit 1 (kanan)
6	b
7	g
8	c
9	dot
10	d
11	e

1.7 PERTANYAAN ANALISIS

1. Jelaskan mekanisme pengaturan pin output sedemikian hingga mengeluarkan nyala lampu sesuai dengan yang diinginkan. Lengkapi dengan penjelasan register yang digunakan!
2. Jelaskan bagaimana pengaturan pin input! Apa pengaruh resistor pull up?
3. Bagaimana proses pengaturan timer sebagai pengganti fungsi delay? Sertakan perhitungannya.
4. Bagaimana proses pengaturan interrupt dengan menggunakan timer? Sertakan perhitungannya.
5. Bagaimana proses pengaturan interupsi tombol pada Arduino Nano? Sertakan register yang terlibat dan fungsinya.

1.8 DAFTAR DATA PRAKTIKUM

1. Percobaan 1.5.1 Output Digital: Tabel Perilaku LED terhadap Waktu
2. Percobaan 1.5.2 Input Digital: Tabel Perilaku LED terhadap Waktu dan Penekanan Tombol
3. Percobaan 1.5.2 Input Digital: Jelaskan pengaruh konsumsi arus pada pin terhadap kinerja Arduino Nano!
4. Percobaan 1.5.2 Input Digital: Grafik Tegangan Pin I/O terhadap Waktu saat Tombol Ditekan
5. Percobaan 1.5.3 Timer: Tabel Perilaku LED terhadap Waktu
6. Percobaan 1.5.4 Interupsi Timer: Tabel Perilaku LED terhadap Waktu
7. Percobaan 1.5.5 Interupsi Tombol: Tabel Perilaku LED terhadap Waktu dan Penekanan Tombol
8. Percobaan 1.6.1 Output Digital: Tabel Perilaku LED terhadap Waktu
9. Percobaan 1.6.2 Input Digital: Tabel Perilaku LED terhadap Waktu dan Penekanan Tombol
10. Percobaan 1.6.2 Input Digital: Grafik Tegangan Pin I/O terhadap Waktu saat Tombol Ditekan
11. Percobaan 1.6.3 Timer: Tabel Perilaku LED terhadap Waktu
12. Percobaan 1.6.4 Interupsi Timer: Tabel Perilaku LED terhadap Waktu
13. Percobaan 1.6.5 Interupsi Tombol: Tabel Perilaku LED terhadap Waktu dan Penekanan Tombol

14. Percobaan 1.6.6 7 Segment: Tabel Perbandingan Nilai yang Ditampilkan Counter terhadap Waktu Sebenarnya (tuliskan data selama 120 detik dengan increment per 10 detik)


INSTITUT TEKNOLOGI BANDUNG
1920

MODUL 2

ESP32

2.1 TUJUAN

- Praktikan memahami datasheet ESP32
- Praktikan mampu membuat aplikasi input dan output pada ESP32 dengan menggunakan bahasa pemrograman C pada ESP-IDF dan Arduino IDE
- Praktikan mampu membuat aplikasi timer dan interrupt pada ESP32 dengan menggunakan bahasa pemrograman C pada ESP-IDF dan Arduino IDE
- Praktikan mampu membuat web server sederhana pada ESP32 menggunakan ESP-IDF dan Arduino IDE

2.2 LANDASAN TEORI

Referensi

Datasheet ESP32:
ttps://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

ESP32 Technical Reference:
https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf
ESP-IDF GPIO API:
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html

ESP-IDF timer API
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/timer.html

ESP-IDF wifi API
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html

ESP-IDF server API
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/esp_http_server.html

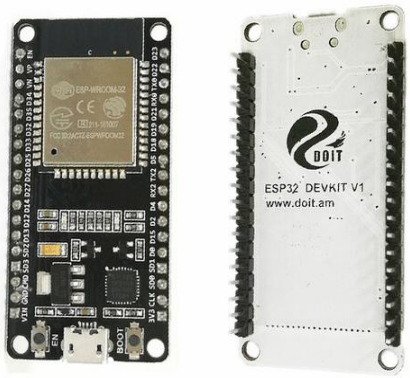
FreeRTOS task.h API:
https://www.freertos.org/a00127.html

2.2.1 ESP32

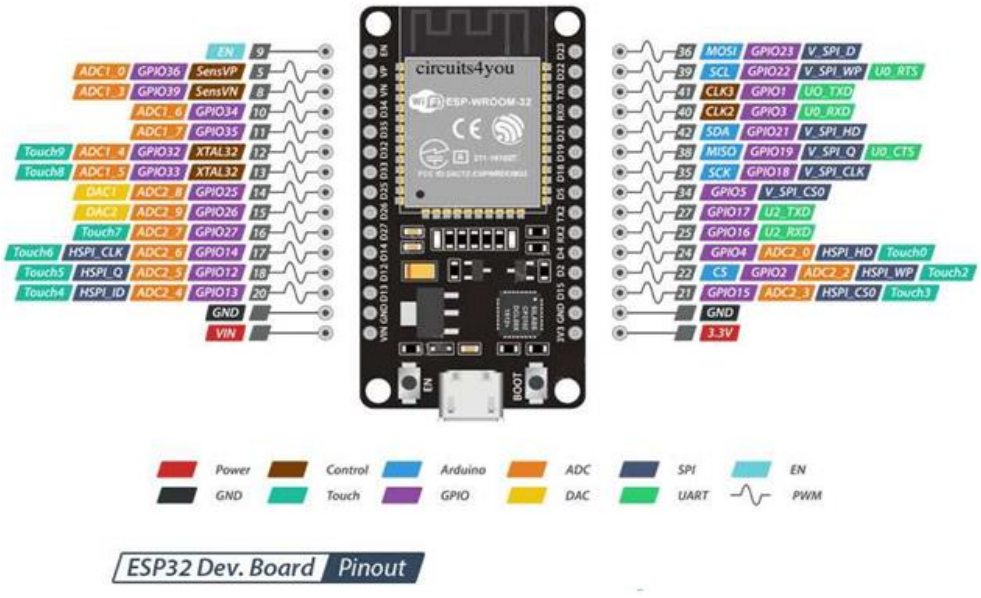
Menurut Espressif Systems, ESP32 adalah chip kombo Wi-Fi dan Bluetooth 2,4 GHz yang dirancang dengan teknologi TSMC ultra-low-power 40 nm. ESP32 dirancang untuk mencapai kinerja daya dan RF terbaik, menunjukkan ketahanan, keserbagunaan, dan keandalan dalam berbagai aplikasi dan skenario daya. FItur utama yang ditawarkan ESP32 adalah sebagai berikut.

- Ultra-Low-Power sehingga cocok untuk aplikasi mobile, wearable electronics, dan Internet-of-Things (IoT).
- Complete Integration untuk aplikasi IoT dengan Wi-Fi dan Bluetooth. ESP32 mengintegrasikan antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, dan modul power management.

Berikut adalah tampilan dari ESP32 versi DOIT (yang akan digunakan pada praktikum) beserta pin-pinnya.



Gambar 2. 1 ESP32 versi DOIT



Gambar 2. 2 ESP32 Dev. Board

2.2.2 Register-Register

Konfigurasi I/O ESP32 dan Interrupt

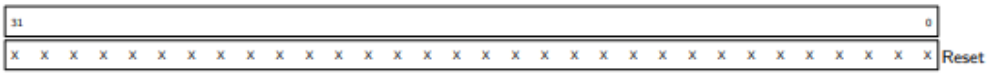
Input dan output pada ESP32 diatur dengan GPIO. Ada 34 pad GPIO pada ESP32: 0-19, 21-23, 25-27, 32-39. Menurut datasheet, pad GPIO 0-19, 21-23, 25-27, 32-33 dapat berfungsi untuk input dan output. Pad GPIO 34-39 hanya untuk input. IO_MUX, RTC IO_MUX, dan matriks GPIO bertanggung jawab untuk merutekan sinyal dari periferal ke pad GPIO. Di bawah ini, akan diperlihatkan beberapa register GPIO matrix sebagai contoh.

DDR

Pada ATmega, mode input (PIN) atau output (PORT) diatur dengan DDR. Pada ESP32, mode input atau output diatur dengan GPIO_ENABLE_REG (jika 1 maka output)

Berikut adalah deskripsi tiap register GPIO matrix untuk Enable (Sumber: ESP32 Technical Reference Hal. 66).

Register 5.7: GPIO_ENABLE_REG (0x0020)



GPIO_ENABLE_REG GPIO0-31 output enable. (R/W)

OUTPUT:

- GPIO_OUT_REG : Mengatur langsung output
- GPIO_OUT_W1TS_REG : Jika bit x diberikan 1, maka bit x pada GPIO_OUT_REG akan berubah menjadi 1
- GPIO_OUT_W1TC_REG : Jika bit x diberikan 1, maka bit x pada GPIO_OUT_REG akan berubah menjadi 0

Catatan: Untuk mengatur output, dapat digunakan 2 cara, yaitu dengan [menggunakan GPIO_OUT_REG langsung] atau [menggunakan GPIO_OUT_W1TS_REG dan GPIO_OUT_W1TC_REG]. Perbedaannya kedua cara ini akan terjadi jika ada multitasking. Penggunaan GPIO_OUT_REG pada multitasking dapat menimbulkan masalah.

Berikut adalah deskripsi tiap register GPIO matrix untuk output (Sumber: ESP32 Technical Reference Hal. 65).

Register 5.1: GPIO_OUT_REG (0x0004)

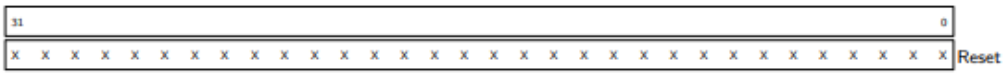
GPIO_OUT_REG GPIO0-31 output value. (R/W)

INPUT:

- GPIO_OUT_REG: Membaca langsung input

Berikut adalah deskripsi tiap register GPIO matrix untuk input (Sumber: ESP32 Technical Reference Hal. 68).

Register 5.14: GPIO_IN_REG (0x003c)



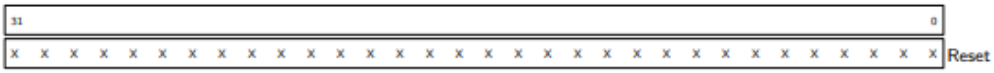
GPIO_IN_REG GPIO0-31 input value. Each bit represents a pad input value, 1 for high level and 0 for low level. (RO)

INTERRUPT:

- GPIO_STATUS_REG: Membaca status interrupt
- GPIO_PINx_REG: Bagian GPIO_PINx_INT_TYPE mengatur tipe interrupt (falling edge, rising edge, dll)

Berikut adalah deskripsi tiap register GPIO matrix untuk interrupt (Sumber: ESP32 Technical Reference Hal. 68).

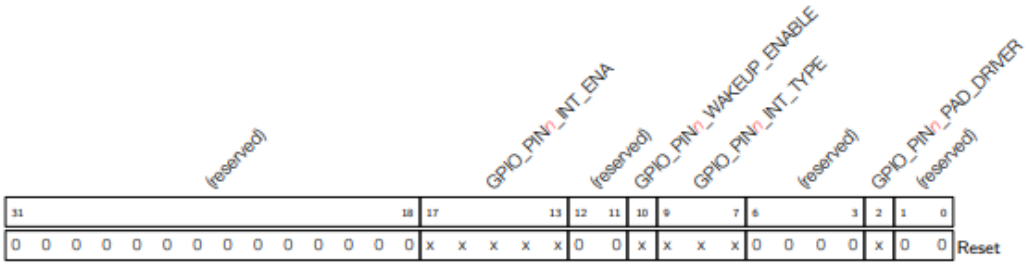
Register 5.16: GPIO_STATUS_REG (0x0044)



GPIO_STATUS_REG GPIO0-31 interrupt status register. Each bit can be either of the two interrupt sources for the two CPUs. The enable bits in GPIO_STATUS_INTERRUPT, corresponding to the 0-4 bits in GPIO_PINn_REG should be set to 1. (R/W)

Berikut adalah deskripsi tiap register GPIO matrix untuk konfigurasi interrupt GPIO pin n (Sumber: ESP32 Technical Reference Hal. 72).

Register 5.30: GPIO_PINn_REG (n: 0-39) (0x88+0x4*n)



GPIO_PINn_INT_ENA Interrupt enable bits for pin n: (R/W)

bit0: APP CPU interrupt enable;
bit1: APP CPU non-maskable interrupt enable;
bit3: PRO CPU interrupt enable;
bit4: PRO CPU non-maskable interrupt enable.

GPIO_PINn_WAKEUP_ENABLE GPIO wake-up enable will only wake up the CPU from Light-sleep. (R/W)

GPIO_PINn_INT_TYPE Interrupt type selection: (R/W)

0: GPIO interrupt disable;
1: rising edge trigger;
2: falling edge trigger;
3: any edge trigger;
4: low level trigger;
5: high level trigger.

GPIO_PINn_PAD_DRIVER 0: normal output; 1: open drain output. (R/W)

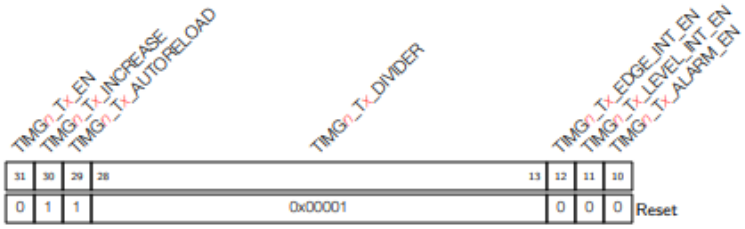
Timer dan Interrupt Timer

Pada ESP32, terdapat 2 jenis timer: 64-bit Timer dan Watchdog Timer. Watchdog Timer adalah timer yang digunakan untuk mereset ESP32, namun timer tersebut tidak akan dibahas pada modul ini. 64-bit Timer adalah timer yang dapat dipakai untuk penggunaan umum seperti yang telah dilakukan dengan ATmega pada Modul 1. Menurut datasheet, ESP32 mempunyai 2 modul Timer (ditandai dengan huruf n) yang mempunyai masing-masing 2 Timer (ditandai dengan huruf x) dengan nama TIMGn_Tx. Berikut adalah fitur 64-bit Timer (Datasheet ESP32 Hal. 25).

- Menggunakan APB clock (APB_CLK, normal 80 MHz) sebagai clock acuan
- 16-bit clock prescaler, dari 2 to 65536
- 64-bit time-base counter
- Configurable up/down time-base counter: incrementing atau decrementing
- Tersedia Interrupt (TIMGn_Tx_INT_T1_INT dan TIMGn_Tx_INT_T0_INT)

Berikut adalah deskripsi tiap register timer (Sumber: ESP32 Technical Reference Hal. 501).

Register 19.1: TIMGn_TxCONFIG_REG (x: 0-1) (0x0+0x24*x)



TIMGn_Tx_EN When set, the timer x time-base counter is enabled. (R/W)

TIMGn_Tx_INCREASE When set, the timer x time-base counter will increment every clock tick. When cleared, the timer x time-base counter will decrement. (R/W)

TIMGn_Tx_AUTORELOAD When set, timer x auto-reload at alarm is enabled. (R/W)

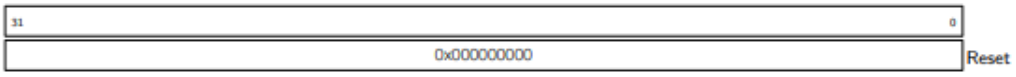
TIMGn_Tx_DIVIDER Timer x clock (Tx_clk) prescale value. (R/W)

TIMGn_Tx_EDGE_INT_EN When set, an alarm will generate an edge type interrupt. (R/W)

TIMGn_Tx_LEVEL_INT_EN When set, an alarm will generate a level type interrupt. (R/W)

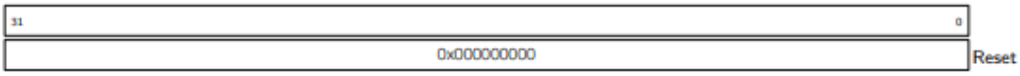
TIMGn_Tx_ALARM_EN When set, the alarm is enabled. This bit is automatically cleared once an alarm occurs. (R/W)

Register 19.4: TIMGn_TxUPDATE_REG (x: 0-1) (0xC+0x24*x)



TIMGn_TxUPDATE_REG Write any value to trigger a timer x time-base counter value update (timer x current value will be stored in registers above). (WO)

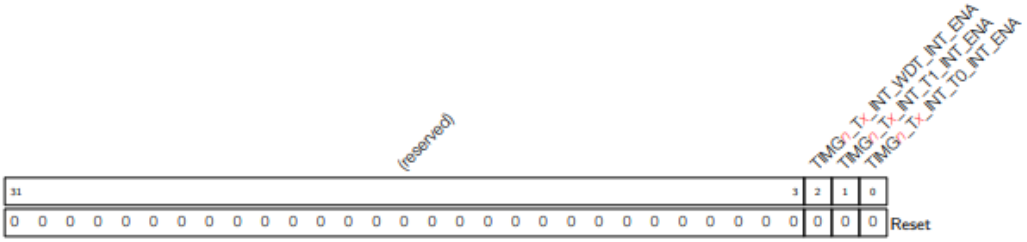
Register 19.9: TIMGn_TxLOAD_REG (x: 0-1) (0x20+0x24*x)



TIMGn_TxLOAD_REG Write any value to trigger a timer x time-base counter reload. (WO)

Berikut adalah deskripsi register timer interrupt (Sumber: ESP32 Technical Reference Hal. 505).

Register 19.18: TIMGn_Tx_INT_ENA_REG (0x0098)

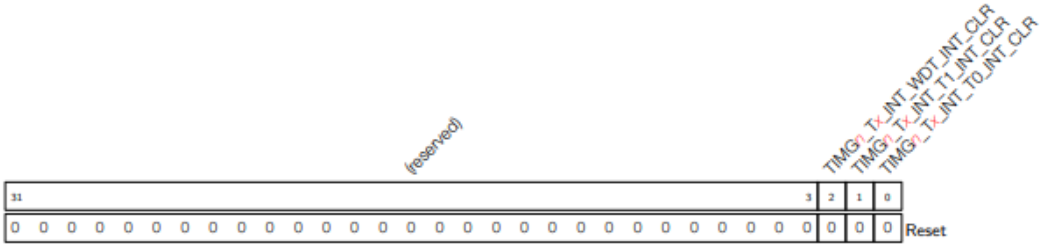


TIMGn_Tx_INT_WDT_INT_ENA The interrupt enable bit for the TIMGn_Tx_INT_WDT_INT interrupt. (R/W) (R/W)

TIMGn_Tx_INT_T1_INT_ENA The interrupt enable bit for the TIMGn_Tx_INT_T1_INT interrupt. (R/W) (R/W)

TIMGn_Tx_INT_T0_INT_ENA The interrupt enable bit for the TIMGn_Tx_INT_T0_INT interrupt. (R/W) (R/W)

Register 19.21: TIMGn_Tx_INT_CLR_REG (0x00a4)



TIMGn_Tx_INT_WDT_INT_CLR Set this bit to clear the TIMGn_Tx_INT_WDT_INT interrupt. (WO)

TIMGn_Tx_INT_T1_INT_CLR Set this bit to clear the TIMGn_Tx_INT_T1_INT interrupt. (WO)

TIMGn_Tx_INT_T0_INT_CLR Set this bit to clear the TIMGn_Tx_INT_T0_INT interrupt. (WO)

2.2.3 ESP-IDF

ESP-IDF adalah IoT Development Framework resmi Espressif untuk SoC seri ESP32 dan ESP32-S. ESP-IDF menyediakan SDK untuk pengembangan aplikasi umum pada platform tersebut menggunakan bahasa pemrograman seperti C dan C ++.

Cara menginstall ESP-IDF disertakan pada lampiran.

2.2.4 API

GPIO.h

Pada praktikum modul 1, telah dicontohkan cara pemrograman mikroprosesor dengan mengubah secara langsung register yang ada. Pada modul 2 ini, akan digunakan API dari gpio.h (library) yang akan melakukan pemrograman register

tersebut secara otomatis ketika kita memanggil fungsi yang ktia inginkan. Berikut adalah beberapa contoh nama fungsi yang akan digunakan dalam modul 2 ini beserta kegunaannya dari dokumentasi Espressif (Sumber: ESP-IDF API Reference).

- **Konfigurasi PIN GPIO**

```
esp_err_t gpio_config(const gpio_config_t * pGPIOConfig)
```

Mengatur konfigurasi umum GPIO (mode, pull-up, pull-down, tipe interrupt).

- **Output**

```
esp_err_t gpio_set_level(gpio_num_t gpio_num, uint32_t level)
```

Mengatur level output dari GPIO.

- **Input**

```
int gpio_get_level(gpio_num_t gpio_num)
```

Mengambil level input GPIO.

- **Interrupt**

```
esp_err_t gpio_install_isr_service(int intr_alloc_flags)
```

Menginstall driver dari GPIO ISR agar GPIO dapat memiliki fitur interrupt.

```
esp_err_t gpio_isr_handler_add(gpio_num_t gpio_num, gpio_isr_t isr_handler, void *args)
```

Menambahkan fitur ISR pada GPIO pin yang dipilih.

task.h

task.h adalah API yang digunakan bersama dengan FreeRTOS.h untuk membuat Real-Time Operating System (RTOS). Penjelasan dan praktikum lebih lanjut mengenai FreeRTOS akan diberikan pada modul 4. Pada modul 2 ini, task.h akan digunakan untuk fungsi delay. Delay yang diapakai adalah vTaskDelay yang penjelasannya adalah sebagai berikut dari dokumentasi FreeRTOS (Sumber: FreeRTOS API Reference).

vTaskDelay

[Task Control]

task. h

```
void vTaskDelay( const TickType_t xTicksToDelay );
```

Mendelay suatu task selama beberapa ticks yang ditentukan pengguna. Tick rate perlu diatur sesuai waktu yang diinginkan.

timer.h

timer.h adalah API yang digunakan untuk mengatur timer pada ESP32 yang akan digunakan. Berikut adalah beberapa contoh nama fungsi timer yang akan digunakan dalam modul 2 ini beserta kegunaannya dari dokumentasi Espressif (Sumber: ESP-IDF API Reference).

- **Konfigurasi Timer**

esp_err_t **timer_init**(timer_group_t *group_num*, timer_idx_t *timer_num*, *const* timer_config_t * *config*)

Inisialisasi timer

esp_err_t **timer_set_counter_value**(timer_group_t *group_num*, timer_idx_t *timer_num*, uint64_t *load_val*)

Mengatur nilai counter pada timer hardware.

esp_err_t **timer_isr_register**(timer_group_t *group_num*, timer_idx_t *timer_num*, void (*fn)(void *), void *arg, int *intr_alloc_flags*, timer_isr_handle_t * *handle*,)

Handler dari register timer interrupt.

esp_err_t **timer_set_alarm_value**(timer_group_t *group_num*, timer_idx_t *timer_num*, uint64_t *alarm_value*)

Mengatur nilai dari alarm timer.

esp_err_t **timer_enable_intr**(timer_group_t *group_num*, timer_idx_t *timer_num*)

Menyalakan interrupt timer.

- **Memulai Timer**

esp_err_t **timer_start**(timer_group_t *group_num*, timer_idx_t *timer_num*)

Memulai counter dari timer hardware.

esp_wifi.h

esp_wifi.h adalah API yang digunakan untuk mengatur dan memonitor fungsi jaringan WiFi pada ESP32. Pengaturan ini termasuk untuk keperluan berikut.

- Station mode (aka STA mode atau WiFi client mode). ESP32 terkoneksi ke access point.
- AP mode (aka Soft-AP mode atau Access Point mode). Station terkoneksi ke ESP32.
- Combined AP-STA mode (ESP32 berperan sebagai access point dan station yang terknoeksi ke access point lainnya).
- Beberapa security modes untuk mode-mdoe diatas (WPA, WPA2, WEP, etc.)
- Scanning untuk access points (active & passive scanning).
- Promiscuous mode untuk monitoring IEEE802.11 WiFi packets.

Berikut adalah beberapa contoh nama fungsi WiFi yang akan digunakan dalam modul 2 ini beserta kegunaannya dari dokumentasi Espressif (Sumber: ESP-IDF API Reference).

```
esp_err_t esp_wifi_init(const wifi_init_config_t * config)
```

Inisialisasi WiFi driver.

```
esp_err_t esp_wifi_set_mode(wifi_mode_t mode)
```

Mengatur mode operasi WiFi.

```
esp_err_t esp_wifi_set_config(wifi_interface_t interface, wifi_config_t
```

Mengatur konfigurasi ESP32 menjadi STA atau AP.

```
esp_err_t esp_wifi_set_storage(wifi_storage_t storage)
```

Set the WiFi API configuration storage type.

Mengatur WiFi

```
esp_err_t esp_wifi_start(void)
```

Memulai WiFi sesuai dengan pengaturan yang diberikan pengguna.

esp_http_server.h

esp_wifi.h adalah API yang digunakan ESP32 dapat menjalankan fungsi web server ringan. Berikut adalah beberapa contoh nama fungsi HTTP yang akan digunakan dalam modul 2 ini beserta kegunaannya dari dokumentasi Espressif (Sumber: ESP-IDF API Reference).

```
esp_err_t httpd_start(httpd_handle_t * handle, const httpd_config_t * config)
```

Menyalakan/memulai web server.

```
esp_err_t httpd_stop(httpd_handle_t handle)
```

Mematikan web server.

```
esp_err_t httpd_register_uri_handler(httpd_handle_t handle, const httpd_uri_t *
```

Meregistrasi uri.

2.3 TUGAS PENDAHULUAN

1. Membuat semua program (source code) yang diperlukan untuk masing-masing percobaan (sertakan keterangan-keterangan penting pada source code menggunakan komentar); Jelaskan masing-masing baris atau bagian kode tersebut.

2.4 ALAT DAN KOMPONEN YANG DIGUNAKAN

- ESP32 (1 buah)
- LED (8 buah)
- Push Button (1 buah)

2.5 MENGGUNAKAN ESP-IDF

Persiapan / Setting awal

- Sebelum melakukan praktikum, praktikan diharuskan untuk menyelesaikan tutorial esp-idf pada lampiran (jika mencapai tutorial nomor 4 maka akan sangat membantu praktikum)

- Menyusun rangkaian yang diperlukan sesuai dengan tugas yang akan dilaksanakan.
- Membuat folder untuk masing-masing tugas (terdapat 6 buah tugas maka terdapat 6 folder). (perhatikan syarat-syarat dalam membuat folder proyek ESP-IDF, dapat dilihat pada tutorial nomor 4.). Folder dibuat dengan format penamaan 251 untuk Output Digital, 252 untuk Input Digital, dan seterusnya secara berurutan.
- Mengisi folder tersebut dengan hasil download dari link berikut https://github.com/espressif/esp-idf-template seperti pada tutorial nomor 4.
- Menyiapkan *environtment* terminal atau command prompt. Windows: Menjalankan ESP-IDF Command Prompt Unix (Linux & macOS/OS X): Masukkan pada terminal
  ```
  cd ~/esp/esp-idf
  ./install.sh
  . $HOME/esp/esp-idf/export.sh
  ```
- Masuk atau *change directory* (cd) ke dalam folder yang diinginkan untuk menjalankan idf.py build / flash / monitor. (build, flash, dan monitor dapat dijalankan secara berurutan dengan sebuah line input seperti `idf.py flash monitor` untuk melakukan *flashing* kemudian membuka *serial communication*)

2.5.1 . Output Digital

1. Membuka main.c pada folder 251.
2. Rangkai 8 buah LED pada *board* ESP32 dengan nomor pin GPIO seperti pada program di bawah ini. (Contoh gambar rangkaian dapat dilihat pada Tugas 2.6.1).
3. Lengkapi dan jalankan contoh program di bawah ini kemudian amati hasilnya.

Tabel 2. 1 Kode Output Digital

```
#include <stdio.h>
// Jika membutuhkan serial.print, cukup printf seperti pada
program C


#include "driver/gpio.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"


#define GPIO_OUTPUT_A    2
#define GPIO_OUTPUT_B    4
#define GPIO_OUTPUT_C    5
#define GPIO_OUTPUT_D    18
#define GPIO_OUTPUT_E    19
#define GPIO_OUTPUT_F    21
#define GPIO_OUTPUT_G    22
#define GPIO_OUTPUT_H    23
```

```
#define GPIO_OUTPUT_PIN_SEL  ((1ULL<<GPIO_OUTPUT_A) |
(1ULL<<GPIO_OUTPUT_B) |
(1ULL<<GPIO_OUTPUT_C)|      (1ULL<<GPIO_OUTPUT_D)|
(1ULL<<GPIO_OUTPUT_E)| (1ULL<<GPIO_OUTPUT_F)|
(1ULL<<GPIO_OUTPUT_G)| (1ULL<<GPIO_OUTPUT_H))

#define DELAY_MS ... // isi waktu delay
const TickType_t xDelay = DELAY_MS / portTICK_PERIOD_MS;

void app_main() {
  gpio_config_t io_conf;
  io_conf.intr_type = ...; // tidak menggunakan interrupt
  io_conf.mode = ...; // mode output
  io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SEL;
  io_conf.pull_down_en = ...; // tidak menggunakan pull down
  io_conf.pull_up_en = ...; // tidak menggunakan pull up
  gpio_config(&io_conf);

  while (1) {
    // Buatlah kondisi dimana 8 buah LED tersebut
(GPIO_OUTPUT_A hingga GPIO_OUTPUT_H) menyala bergantian
(menggunakanan gpio_set_level())setiap 0,5 detik dengan
menggunakan vTaskDelay dan variabel xDelay di atas.
  }
}
```

4. Apabila kondisi 8 buah LED tersebut menyala bergantian sekitar setiap 500 milidetik sesuai dengan nilai pada DELAY_MS, maka percobaan yang dilakukan benar.
5. **Modifikasi program sehingga LED menyala bergeser secara bergantian. Pertama-tama LED A menyala dan semua LED lainnya mati, kemudian LED B menyala dan semua LED lainnya mati, dst**.

2.5.2 . Input Digital

1. Membuka main.c pada folder 252.
2. Menggunakan rangkaian pada Tugas 2.5.1, tambahkan *push button* dengan 1 kaki tersambung pada *ground* (GND), dan 1 kaki lainnya tersambung pada GPIO15.
3. Lengkapi dan jalankan contoh program di bawah ini kemudian amati hasilnya.

Tabel 2. 2 Kode Input Digital

```
#include <stdio.h>
#include "driver/gpio.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
```

```
#define GPIO_OUTPUT_A    2
#define GPIO_OUTPUT_B    4
#define GPIO_OUTPUT_C    5
#define GPIO_OUTPUT_D    18
#define GPIO_OUTPUT_E    19
#define GPIO_OUTPUT_F    21
#define GPIO_OUTPUT_G    22
#define GPIO_OUTPUT_H    23
#define GPIO_OUTPUT_PIN_SEL  ((1ULL<<GPIO_OUTPUT_A) |
(1ULL<<GPIO_OUTPUT_B) |
(1ULL<<GPIO_OUTPUT_C)|      (1ULL<<GPIO_OUTPUT_D)|
(1ULL<<GPIO_OUTPUT_E)| (1ULL<<GPIO_OUTPUT_F)|
(1ULL<<GPIO_OUTPUT_G)| (1ULL<<GPIO_OUTPUT_H))

#define GPIO_INPUT_PB    15
#define GPIO_INPUT_PIN_SEL  (1ULL<<GPIO_INPUT_PB)

#define DELAY_MS 200
const TickType_t xDelay = DELAY_MS / portTICK_PERIOD_MS;

void app_main() {
  gpio_config_t io_conf;
  io_conf.intr_type = ...;
  io_conf.mode = ...;
  io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SEL;
  io_conf.pull_down_en = ...;
  io_conf.pull_up_en = ...;
  gpio_config(&io_conf);

  io_conf.pin_bit_mask = GPIO_INPUT_PIN_SEL;
  io_conf.mode = ...; // mode input
  io_conf.pull_up_en = ...; // menggunakan pull up
  gpio_config(&io_conf);

  while (1) {
    if (gpio_get_level(GPIO_INPUT_PB) == 0) {
      // masukkan potongan program LED menyala bergantian
dari tugas 2.5.1 Output Digital
    }
    // vTaskDelay(1); // Jika terdapat error setelah
melakukan flash (saat menggunakan IDF Monitor), coba untuk
uncomment line ini terlebih dahulu

  }
}
```

4. Apabila kondisi 8 buah LED tersebut menyala bergeser ketika button ditekan, maka percobaan yang dilakukan benar.
5. Amati perilaku push button ketika ditekan dengan osiloskop.

6. Lakukan pengukuran tegangan pin I/O terhadap GND, LED terhadap GND, serta arus pada salah satu LED yang menyala.
7. Amati tegangan pada pin 5V terhadap setiap jumlah nyala LED. Misal, tegangan pin 5V pada saat 1 buah LED menyala sampai 8 buah LED menyala.
8. **Modifikasi program sehingga beroperasi dengan *push button pull-down* menggunakan internal *pull-down resistor* pada board ESP32.**
9. Amati perilaku push button ketika ditekan dengan osiloskop untuk mode *pull-down*.

2.5.3 . Timer

1. Membuka main.c pada folder 253
2. Gunakan rangkaian pada Tugas 2.5.1
3. Lengkapi dan jalankan contoh program di bawah ini kemudian amati hasilnya.

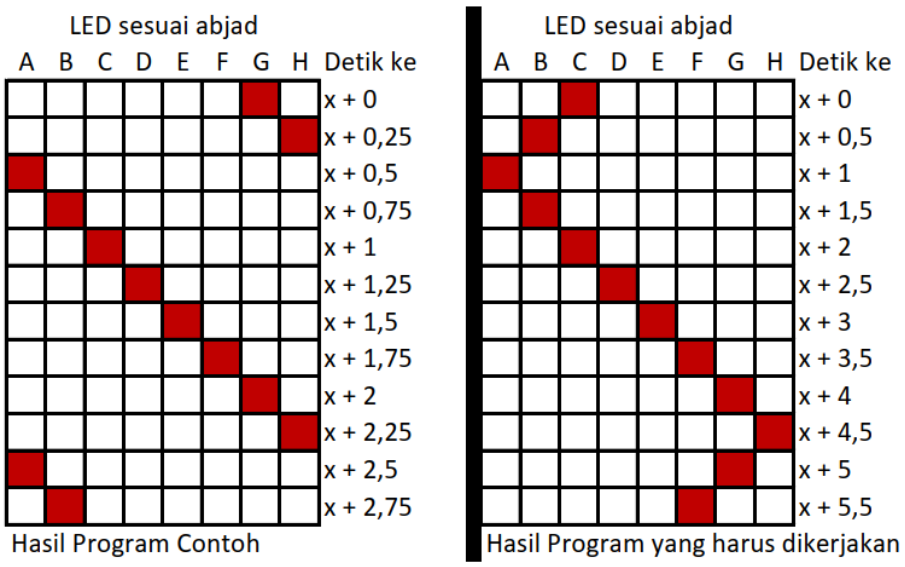
Tabel 2. 3 Kode Timer

```
#include <stdio.h>
#include "driver/gpio.h"
#include "driver/timer.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

#define GPIO_OUTPUT_A    2
#define GPIO_OUTPUT_B    4
#define GPIO_OUTPUT_C    5
#define GPIO_OUTPUT_D    18
#define GPIO_OUTPUT_E    19
#define GPIO_OUTPUT_F    21
#define GPIO_OUTPUT_G    22
#define GPIO_OUTPUT_H    23
#define GPIO_OUTPUT_PIN_SEL  ((1ULL<<GPIO_OUTPUT_A) |
(1ULL<<GPIO_OUTPUT_B) | (1ULL<<GPIO_OUTPUT_C)|
(1ULL<<GPIO_OUTPUT_D)| (1ULL<<GPIO_OUTPUT_E)|
(1ULL<<GPIO_OUTPUT_F)| (1ULL<<GPIO_OUTPUT_G)|
(1ULL<<GPIO_OUTPUT_H))

#define TIMER_DIVIDER         16
#define TIMER_SCALE           (TIMER_BASE_CLK /
TIMER_DIVIDER)
#define DELAY_S               0.25
#define NUMBER_OF_LED         8
#define TIMER1_INTERVAL_SEC   (DELAY_S * NUMBER_OF_LED)

void app_main(void) {
  gpio_config_t io_conf;
  io_conf.intr_type = ...;
  io_conf.mode = ...;
```

```
  io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SEL;
  io_conf.pull_down_en = ...;
  io_conf.pull_up_en = ...;
  gpio_config(&io_conf);

  // Timer menghitung ke atas, nanti baru di start, pakai
alarm, namun tanpa reload counter value setelah alarm event.
  timer_config_t config = {
      .divider = TIMER_DIVIDER,
      .counter_dir = ...,
      .counter_en = ...,
      .alarm_en = ...,
      .auto_reload = ...,
  };
  // gunakan timer group dan hardware timer yang valid
  timer_init(..., ..., &config);
  timer_set_counter_value(..., ..., 0x00000000ULL);
  timer_start(..., ...);

  // Silahkan melengkapi potongan kode berikut ini untuk
membuat program LED menyala bergeser setiap 250ms dan
berulang mulai dari LED paling awal.
  // Jika ingin membuat potongan kode sendiri sangat
dipersilahkan (sekaligus dapat mengubah konfigurasi timer di
atas)
  int count = -1;
  double current_time_sec = 0, last_time_sec = 0,
last_reset_time = 0;
  while (1) {
    timer_get_counter_time_sec(..., ..., &current_time_sec);
    if (current_time_sec - last_time_sec > DELAY_S) {
      count++;
      last_time_sec = current_time_sec;
    } else if (current_time_sec - last_reset_time >
TIMER1_INTERVAL_SEC) {
      count = -1;
      last_reset_time = current_time_sec;
    }
    // Buatlah kondisi dimana 8 buah LED tersebut
(GPIO_OUTPUT_A hingga GPIO_OUTPUT_H) menyala bergantian hanya
menggunakan variabel count dengan fungsi/API
gpio_set_level(). Total hanya memerlukan 8 line tambahan
    vTaskDelay(1);
  }
}
```

4. Apabila kondisi 8 buah LED tersebut menyala bergeser setiap 250 ms dan berulang mulai dari LED paling awal (pola perulangan hanya dari kiri ke kanan seperti pada gambar di bawah kiri), maka percobaan yang dilakukan benar.

(urutan LED menyala ⇒ LED_0 ON → LED_1 ON → LED_2 ON → ... → LED_7 ON → LED_0 ON → LED_1 ON → dst)



Gambar 2. 3 Hasil pergerakan LED (kiri) program contoh instruksi nomor 4 dan (kanan) program hasil modifikasi praktikan instruksi nomor 5

5. **Modifikasi program sehingga waktu delay antar lampu bernilai 0,5 detik dan juga pada program yang sama, lakukan modifikasi sehingga LED memiliki pola perulangan dari kiri ke kanan, kanan ke kiri, kiri ke kanan, dan seterusnya seperti pada gambar diatas kanan.**

   (urutan LED menyala ⇒ LED_0 ON → LED_1 ON → LED2 ON → ... → LED_6 ON → LED7 ON → LED_6 ON → LED_5 ON → ... → LED_1 ON → LED_0 ON → LED_1 ON → dst)

2.5.4 . Interupsi Timer

1. Membuka main.c pada folder 254
2. Gunakan rangkaian pada Tugas 2.5.1
3. Lengkapi dan jalankan contoh program di bawah ini kemudian amati hasilnya.

Tabel 2. 4 Kode Interupsi Timer

```
#include <stdio.h>
#include "driver/gpio.h"
#include "driver/timer.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

#define GPIO_OUTPUT_A    2
#define GPIO_OUTPUT_B    4
```

```
#define GPIO_OUTPUT_C    5
#define GPIO_OUTPUT_D    18
#define GPIO_OUTPUT_E    19
#define GPIO_OUTPUT_F    21
#define GPIO_OUTPUT_G    22
#define GPIO_OUTPUT_H    23
#define GPIO_OUTPUT_PIN_SEL  ((1ULL<<GPIO_OUTPUT_A) |
(1ULL<<GPIO_OUTPUT_B) | (1ULL<<GPIO_OUTPUT_C)|
(1ULL<<GPIO_OUTPUT_D)| (1ULL<<GPIO_OUTPUT_E)|
(1ULL<<GPIO_OUTPUT_F)| (1ULL<<GPIO_OUTPUT_G)|
(1ULL<<GPIO_OUTPUT_H))

#define TIMER_USED             ... // Isi dengan Hardware
Timer 0
#define TIMER_DIVIDER          16
#define TIMER_SCALE            (TIMER_BASE_CLK /
TIMER_DIVIDER)
#define DELAY_S                (1.0)
#define NUMBER_OF_LED          8

int led_state= 0;
// isi dengan attribut yang membuat interrupt hanya dapat
dipanggil pada IRAM/ROM dan juga isi nama fungsi
interruptnya.
void ... ...(void* para) {
  // semua timer group pada interrupt harus sama dengan
app_main
  timer_spinlock_take(...);
  int timer_idx = (int)para;
  uint32_t timer_intr =
timer_group_get_intr_status_in_isr(...);

  if (timer_intr & TIMER_INTR_T0) {
    timer_group_clr_intr_status_in_isr(..., TIMER_0);
  } else if (timer_intr & TIMER_INTR_T1) {
    timer_group_clr_intr_status_in_isr(..., TIMER_1);
  }

  // Lengkapi dengan potongan kode yang merubah kondisi LED
(menyala menjadi mati, mati menjadi menyala) setiap kali
interrupt ini dipanggil.
  // Dapat juga dilengkapi dengan potongan kode yang
memprogram perilaku LED selain berkedip

  timer_group_enable_alarm_in_isr(..., timer_idx);
  timer_spinlock_give(...);
}

void app_main(void) {
  gpio_config_t io_conf;
  io_conf.intr_type = ...;
  io_conf.mode = ...;
```

```
  io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SEL;
  io_conf.pull_down_en = ...;
  io_conf.pull_up_en = ...;
  gpio_config(&io_conf);

  // Buat timer config yang melakukan load counter value
setelah alarm event
  timer_config_t config = {
      .divider = ...,
      .counter_dir = ...,
      .counter_en = ...,
      .alarm_en = ...,
      .auto_reload = ...,
  };
  timer_init(..., TIMER_USED, &config);
  timer_set_counter_value(..., TIMER_USED, 0x00000000ULL);
  timer_set_alarm_value(..., TIMER_USED, DELAY_S *
TIMER_SCALE);
  timer_enable_intr(..., TIMER_USED);
  // jangan lupa isi dengan nama fungsi interrupt dan isi
intr_alloc_flags agar interrupt hanya dapat dipanggil pada
IRAM/ROM
  timer_isr_register(..., TIMER_USED, ...,
    (void*)TIMER_USED, ..., NULL);
  timer_start(..., TIMER_USED);

  while (1) {
    vTaskDelay(1);
  };
}
```

4. Apabila kondisi 8 buah LED tersebut berkedip dengan periode 2 detik, maka percobaan yang dilakukan benar.
5. **Modifikasi program sehingga LED memiliki perilaku yang berbeda selain menyala dan mati (perilaku berkedip boleh diganti dengan perilaku lain atau dapat menambah jumlah perilaku).**
6. **Pada program yang sama dengan nomor 5, *hardware timer* yang digunakan juga harus dimodifikasi sehingga menggunakan TIMER_1 dan bukan TIMER_0 dengan waktu *delay* pergantian perilaku menjadi 3 detik (hint: perhatikan *macro*).**

2.5.5 . Interupsi Tombol

1. Membuka main.c pada folder 255
2. Gunakan rangkaian pada Tugas 2.5.2
3. Lengkapi dan jalankan contoh program di bawah ini kemudian amati hasilnya.

Tabel 2. 5 Kode Interupsi Tombol

```
#include <stdio.h>
#include "driver/gpio.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

#define GPIO_OUTPUT_A    2
#define GPIO_OUTPUT_B    4
#define GPIO_OUTPUT_C    5
#define GPIO_OUTPUT_D    18
#define GPIO_OUTPUT_E    19
#define GPIO_OUTPUT_F    21
#define GPIO_OUTPUT_G    22
#define GPIO_OUTPUT_H    23
#define GPIO_OUTPUT_PIN_SEL  ((1ULL<<GPIO_OUTPUT_A) |
(1ULL<<GPIO_OUTPUT_B) |
(1ULL<<GPIO_OUTPUT_C)|       (1ULL<<GPIO_OUTPUT_D)|
(1ULL<<GPIO_OUTPUT_E)| (1ULL<<GPIO_OUTPUT_F)|
(1ULL<<GPIO_OUTPUT_G)| (1ULL<<GPIO_OUTPUT_H))

#define GPIO_INPUT_PB    15
#define GPIO_INPUT_PIN_SEL  (1ULL<<GPIO_INPUT_PB)

#define ESP_INTR_FLAG_DEFAULT 0

int on_led_count = 0;

// isi dengan attribut yang membuat interrupt hanya dapat
dipanggil pada IRAM/ROM dan juga isi nama fungsi
interruptnya.
static void ... ...(void* arg) {
  // tulis potongan kode untuk interrupt
  ...
}

void app_main() {
  gpio_config_t io_conf;
  io_conf.intr_type = ...;
  io_conf.mode = ...;
  io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SEL;
  io_conf.pull_down_en = ...;
  io_conf.pull_up_en = ...;
  gpio_config(&io_conf);

  io_conf.pin_bit_mask = GPIO_INPUT_PIN_SEL;
  io_conf.mode = ...;
  io_conf.intr_type = ...; // Isi dengan tipe interrupt
(rising, falling, both, dll) agar terdapat interrupt pada
kondisi push button pertama kali ditekan dan bukan pada
kondisi lainnya.
  io_conf.pull_up_en = 1;
  gpio_config(&io_conf);
```

```
  // Tambahkan potongan kode gpio_install_isr_service()
dengan ESP_INTR_FLAG_DEFAULT dan gpio_isr_handler_add(),
jangan lupa untuk mengisi parameter kedua fungsi/API tersebut

  while (1) {
    // Buatlah potongan kode agar LED dapat menyala bergeser
ketika button ditekan, gabungkan dengan fungsionalitas
interrupt
    // Kode dapat sepenuhnya berada pada interrupt atau dapat
juga dikombinasikan interrupt dan main loop. Dengan kata
lain, interrupt harus selalu digunakan.
    ...
    vTaskDelay(1);
  }
}
```

4. Apabila kondisi 8 buah LED tersebut menyala bergeser ketika button ditekan, maka percobaan yang dilakukan benar.
5. **Modifikasi program sehingga interrupt terjadi ketika terjadi *rising edge* (POSEDGE).**

2.5.6 . Wi-Fi dan IoT

1. Membuka main.c pada folder 256
2. Hubungkan sebuah LED pada GPIO 22
3. Lengkapi dan jalankan contoh program di bawah ini, kemudian *connect* ke WiFi ESP32 dengan ssid "myssid" dan password "mypassword" melalui perangkat *smartphone* anda. Masuk ke browser lalu ketik 192.168.4.1 dan amati hasilnya pada GPIO 22 dan IDF Monitor ketika tombol ON/OFF pada *webpage* ditekan seperti pada gambar di bawah.

Tabel 2. 6 Kode Wi-Fi dan IoT

```
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "driver/gpio.h"

#include "lwip/err.h"
#include "lwip/sys.h"
#include <esp_http_server.h>

// SSID tidak perlu diubah, jika ingin mengubah, sebaiknya jangan
mengubah menjadi ssid yang sama dengan yang sudah ada pada
jangkauan.
#define EXAMPLE_ESP_WIFI_SSID      "myssid"
```

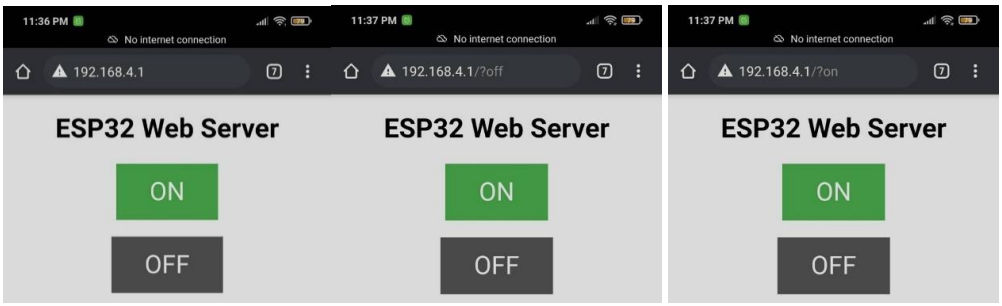
```
#define EXAMPLE_ESP_WIFI_PASS      "mypassword"
#define EXAMPLE_ESP_WIFI_CHANNEL   1
#define EXAMPLE_MAX_STA_CONN       4
#define LED_PIN                    22
#define GPIO_OUTPUT_PIN_SEL        (1ULL<<LED_PIN)

static const char* TAG_WIFI = "wifi softAP";
static const char* TAG_SERVER = "webserver";

static void wifi_event_handler(void* arg, esp_event_base_t
event_base,
  int32_t event_id, void* event_data) {
  if (event_id == WIFI_EVENT_AP_STACONNECTED) {
    wifi_event_ap_staconnected_t* event =
(wifi_event_ap_staconnected_t*)event_data;
    ESP_LOGI(TAG_WIFI, "station "MACSTR" join, AID=%d",
      MAC2STR(event->mac), event->aid);
  } else if (event_id == WIFI_EVENT_AP_STADISCONNECTED) {
    wifi_event_ap_stadisconnected_t* event =
(wifi_event_ap_stadisconnected_t*)event_data;
    ESP_LOGI(TAG_WIFI, "station "MACSTR" leave, AID=%d",
      MAC2STR(event->mac), event->aid);
  }
}

void wifi_init_softap(void) {
  ESP_ERROR_CHECK(esp_netif_init());
  ESP_ERROR_CHECK(esp_event_loop_create_default());
  esp_netif_create_default_wifi_ap();

  wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
  ESP_ERROR_CHECK(esp_wifi_init(&cfg));

  ESP_ERROR_CHECK(esp_event_handler_instance_register(WIFI_EVENT,
    ESP_EVENT_ANY_ID,
    &wifi_event_handler,
    NULL,
    NULL));

  wifi_config_t wifi_config = {
      .ap = {
          .ssid = EXAMPLE_ESP_WIFI_SSID,
          .ssid_len = strlen(EXAMPLE_ESP_WIFI_SSID),
          .channel = EXAMPLE_ESP_WIFI_CHANNEL,
          .password = EXAMPLE_ESP_WIFI_PASS,
          .max_connection = EXAMPLE_MAX_STA_CONN,
          .authmode = WIFI_AUTH_WPA_WPA2_PSK
      },
  };
  if (strlen(EXAMPLE_ESP_WIFI_PASS) == 0) {
    wifi_config.ap.authmode = WIFI_AUTH_OPEN;
  }
```

```
  ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_AP));
  ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_AP,
&wifi_config));
  ESP_ERROR_CHECK(esp_wifi_start());

  ESP_LOGI(TAG_WIFI, "wifi_init_softap finished. SSID:%s
password:%s channel:%d",
    EXAMPLE_ESP_WIFI_SSID, EXAMPLE_ESP_WIFI_PASS,
EXAMPLE_ESP_WIFI_CHANNEL);
}

static esp_err_t hello_get_handler(httpd_req_t* req) {
  char* buf;
  size_t buf_len;

  buf_len = httpd_req_get_hdr_value_len(req, "Host") + 1;
  if (buf_len > 1) {
    buf = malloc(buf_len);
    if (httpd_req_get_hdr_value_str(req, "Host", buf, buf_len) ==
ESP_OK) {
      ESP_LOGI(TAG_SERVER, "Found header => Host: %s", buf);
    }
    free(buf);
  }

  buf_len = httpd_req_get_hdr_value_len(req, "Test-Header-2") +
1;
  if (buf_len > 1) {
    buf = malloc(buf_len);
    if (httpd_req_get_hdr_value_str(req, "Test-Header-2", buf,
buf_len) == ESP_OK) {
      ESP_LOGI(TAG_SERVER, "Found header => Test-Header-2: %s",
buf);
    }
    free(buf);
  }

  buf_len = httpd_req_get_hdr_value_len(req, "Test-Header-1") +
1;
  if (buf_len > 1) {
    buf = malloc(buf_len);
    if (httpd_req_get_hdr_value_str(req, "Test-Header-1", buf,
buf_len) == ESP_OK) {
      ESP_LOGI(TAG_SERVER, "Found header => Test-Header-1: %s",
buf);
    }
    free(buf);
  }

  buf_len = httpd_req_get_url_query_len(req) + 1;
  if (buf_len > 1) {
```

```
    buf = malloc(buf_len);
    if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK)
{
      ESP_LOGI(TAG_SERVER, "Found URL query => %s", buf);
      if (strcmp(buf, "on") == 0) {
        gpio_set_level(LED_PIN, 1);
      } else {
        gpio_set_level(LED_PIN, 0);
      }
      char param[32];
      if (httpd_query_key_value(buf, "query1", param,
sizeof(param)) == ESP_OK) {
        ESP_LOGI(TAG_SERVER, "Found URL query parameter =>
query1=%s", param);
      }
      if (httpd_query_key_value(buf, "query3", param,
sizeof(param)) == ESP_OK) {
        ESP_LOGI(TAG_SERVER, "Found URL query parameter =>
query3=%s", param);
      }
      if (httpd_query_key_value(buf, "query2", param,
sizeof(param)) == ESP_OK) {
        ESP_LOGI(TAG_SERVER, "Found URL query parameter =>
query2=%s", param);
      }
    }
    free(buf);
  }

  httpd_resp_set_hdr(req, "Custom-Header-1", "Custom-Value-1");
  httpd_resp_set_hdr(req, "Custom-Header-2", "Custom-Value-2");

  const char* resp_str = (const char*)req->user_ctx;
  httpd_resp_send(req, resp_str, strlen(resp_str));

  if (httpd_req_get_hdr_value_len(req, "Host") == 0) {
    ESP_LOGI(TAG_SERVER, "Request headers lost");
  }
  return ESP_OK;
}

static const httpd_uri_t hello = {
    .uri = "/",
    .method = HTTP_GET,
    .handler = hello_get_handler,
    .user_ctx = "<!DOCTYPE html><html><head><meta
name=\"viewport\" content=\"width=device-width, initial-
scale=1\"><link rel=\"icon\" href=\"data:,\"><style>html { font-
family: Helvetica; display: inline-block; margin: 0px auto; text-
align: center;}.button { background-color: #4CAF50; border: none;
color: white; padding: 16px 40px;text-decoration: none; font-
size: 30px; margin: 2px; cursor: pointer;}.button2 {background-
```

```
color: #555555;}</style></head><body><h1>ESP32 Web
Server</h1><p><a href=\"?on\"><button
class=\"button\">ON</button></a></p><p><a href=\"?off\"><button
class=\"button button2\">OFF</button></a></p>"
};

esp_err_t http_404_error_handler(httpd_req_t* req,
httpd_err_code_t err) {
  if (strcmp("/", req->uri) == 0) {
    httpd_resp_send_err(req, HTTPD_404_NOT_FOUND, "/ URI is not
available");
    return ESP_OK;
  }
  httpd_resp_send_err(req, HTTPD_404_NOT_FOUND, "Some 404 error
message");
  return ESP_FAIL;
}

static httpd_handle_t start_webserver(void) {
  httpd_handle_t server = NULL;
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();

  ESP_LOGI(TAG_SERVER, "Starting server on port: '%d'",
config.server_port);
  if (httpd_start(&server, &config) == ESP_OK) {
    ESP_LOGI(TAG_SERVER, "Registering URI handlers");
    httpd_register_uri_handler(server, &hello);
    return server;
  }

  ESP_LOGI(TAG_SERVER, "Error starting server!");
  return NULL;
}

void app_main(void) {
  // jangan lupa diisi
  gpio_config_t io_conf;
  io_conf.intr_type = ...;
  io_conf.mode = ...;
  io_conf.pin_bit_mask = ...;
  io_conf.pull_down_en = ...;
  io_conf.pull_up_en = ...;
  gpio_config(&io_conf);

  esp_err_t ret = nvs_flash_init();
  if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret ==
ESP_ERR_NVS_NEW_VERSION_FOUND) {
    ESP_ERROR_CHECK(nvs_flash_erase());
    ret = nvs_flash_init();
  }
  ESP_ERROR_CHECK(ret);
```

```
  ESP_LOGI(TAG_WIFI, "ESP_WIFI_MODE_AP");
  wifi_init_softap();

  static httpd_handle_t server = NULL;
  server = start_webserver();
}
```



Gambar 2. 4 *Webpage* saat (kiri) baru diakses (tengah) tombol *off* ditekan (kanan) tombol *on* ditekan

4. **Modifikasi program sehingga *webpage* dapat mengatur kondisi (nyala/mati) 2 buah LED (GPIO yang digunakan dibebaskan kepada praktikan).**

2.6 MENGGUNAKAN ARDUINO IDE

2.6.1 . Output Digital

1. Buka Arduino IDE. Pastikan konfigurasi dari Arduino IDE sudah bisa menggunakan ESP32.

2. Buat rangkaian sederhana dengan 8 buah LED. Contoh rangkaian ESP32 dengan LED dapat dilihat pada gambar di bawah:

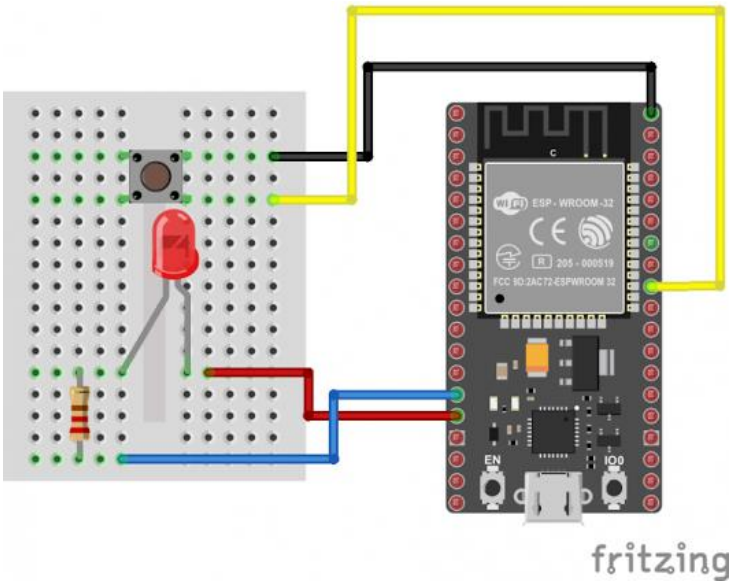
Gambar 2. 5 Contoh Rangkaian ESP32 dengan LED

Buat program sehingga dapat menampilkan nyala LED secara berurutan dengan delay 0,5 detik (tips: atur posisi **DigitalWrite** dan *delay* serta atur nilai pada *delay*).

2.6.2 . Input Digital

1. Buka Arduino IDE. Pastikan konfigurasi dari Arduino IDE sudah bisa menggunakan ESP32.

2. Buat rangkaian ESP32 dengan input berupa *push button* dan LED 8 buah. Untuk contoh konfigurasi dapat mengikuti contoh dibawah ini (sesuaikan dengan jumlah LED):



Gambar 2. 6 Contoh konfigurasi rangkaian ESP32 dengan input berupa *push button* dan LED

2. Jalankan contoh program di bawah ini (sesuaikan pin dengan pin pada konfigurasi yang praktikan buat), amati hasilnya.

4. Kemudian ubah kode di *line* 17 (PinMode) menjadi INPUT_PULLDOWN. *Upload* program lalu amati hasilnya.

5. Modifikasi program sehingga dapat menampilkan nyala LED secara berurutan dengan delay 0,5 detik (tips: atur posisi **DigitalWrite** dan *delay* serta atur nilai pada *delay*).

Note : Saat membuat konfigurasi pin untuk internal *pull up* dan *pull down* hindari penggunaan pin 0-6 (tidak dapat *booting* bila digunakan sebagai input). Untuk *pull up*, hindari penggunaan pin 34-39 (tidak ada internal *pull up* resistor) dan untuk *pull down* hindari pin 0-3 (tidak bekerja untuk input *pull down*).

2.6.3 . Timer

1. Buka Arduino IDE. Pastikan konfigurasi dari Arduino IDE sudah bisa menggunakan ESP32.

2. Buat program dengan perilaku LED seperti pada Tugas 2.5.3

2.6.4 . Interupsi Timer

1. Buka Arduino IDE. Pastikan konfigurasi dari Arduino IDE sudah bisa menggunakan ESP32.

2. Buat program dengan perilaku LED seperti pada Tugas 2.5.4

2.6.5 . Interupsi Tombol

1. Buka Arduino IDE. Pastikan konfigurasi dari Arduino IDE sudah bisa menggunakan ESP32.

2. Buat rangkaian yang sama dengan rangkaian pada tugas **Input Digital.**

3. Buat program dengan perilaku LED seperti pada Tugas 2.5.5
4. Ubah mode *interrupt* menjadi perilaku lain. Terdapat 5 mode *interrupt*, yaitu:
    a. LOW
    b. HIGH
    c. CHANGE
    d. FALLING
    e. RISING

    Amati hasilnya.

    Note : Pada kode diatas, ditambahkan pula kondisi *debouncing*. Lakukan analisis terkait *debouncing* ini di laporan praktikan.

2.6.6 . Wi-Fi dan IoT

Langkah percobaannya adalah sebagai berikut:

1. Buka Arduino IDE. Pastikan konfigurasi dari Arduino IDE sudah bisa menggunakan ESP32.

2. Buat rangkaian LED sederhana seperti pada tugas **Output Digital** tetapi cukup dengan 1 buah LED saja (konfigurasi pin diatur oleh praktikan).

3. Jalankan contoh program di bawah ini (sesuaikan pin dengan pin pada konfigurasi yang praktikan buat). Perhatikan pada bagian **ssid** dan **password**, isikan dengan nama WiFi / *Hotspot* dari praktikan lalu *password* dari WiFi / *Hotspot* praktikan. Lalu, isikan konfigurasi pin LED pada *line* 18 (**const int output = ..**). *Upload* program lalu amati hasilnya.

Tabel 2. 7 Kode Wi-Fi dan IoT

```
// Load Wi-Fi library
#include <WiFi.h>

// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Auxiliar variables to store the current output state
String outputState = "off";

// Assign output variables to GPIO pins
const int output = ;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
  Serial.begin(115200);
  // Initialize the output variables as outputs
  pinMode(output, OUTPUT);
  // Set outputs to LOW
  digitalWrite(output, LOW);

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop(){
  WiFiClient client = server.available();    // Listen for
incoming clients

  if (client) {                             // If a new client
connects,
      currentTime = millis();
      previousTime = currentTime;
```

```
      Serial.println("New Client.");             // print a
message out in the serial port
      String currentLine = "";                   // make a
String to hold incoming data from the client
      while (client.connected() && currentTime - previousTime
<= timeoutTime) {  // loop while the client's connected
      currentTime = millis();
      if (client.available()) {          // if there's bytes
to read from the client,
      char c = client.read();             // read a byte,
then
      Serial.write(c);                    // print it out the
serial monitor
      header += c;
      if (c == '\n') {                 // if the byte is a
newline character
            // if the current line is blank, you got two
newline characters in a row.
            // that's the end of the client HTTP request, so
send a response:
            if (currentLine.length() == 0) {
            // HTTP headers always start with a response code
(e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's
coming, then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // turns the GPIOs on and off
            if (header.indexOf("GET /26/on") >= 0) {
              Serial.println("GPIO on");
            outputState = "on";
              digitalWrite(output, HIGH);
            } else if (header.indexOf("GET /26/off") >= 0) {
              Serial.println("GPIO off");
            outputState = "off";
              digitalWrite(output, LOW);
            }

            // Display the HTML web page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta    name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
            client.println("<link               rel=\"icon\"
href=\"data:,\">");
            // CSS to style the on/off buttons
            // Feel free to change the background-color and
font-size attributes to fit your preferences
            client.println("<style>html    {    font-family:
Helvetica; display: inline-block; margin: 0px auto; text-
align: center;}");
            client.println(".button    {    background-color:
#4CAF50; border: none; color: white; padding: 16px 40px;");
            client.println("text-decoration:   none;   font-
size: 30px; margin: 2px; cursor: pointer;}");
            client.println(".button2      {background-color:
#555555;}</style></head>");
```

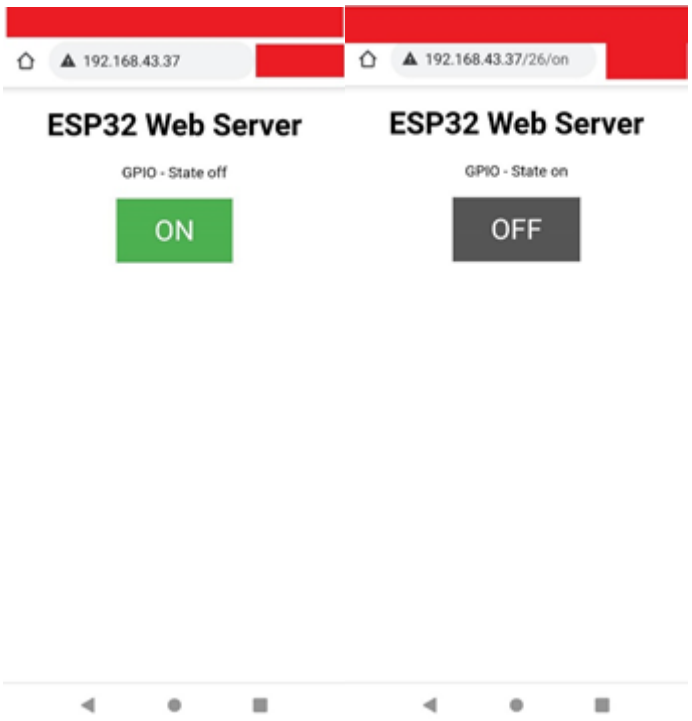
```
            // Web Page Heading
            client.println("<body><h1>ESP32                Web
Server</h1>");

            // code for interacting with LED Pin
            // Display current state, and ON/OFF buttons for
GPIO
            client.println("<p>GPIO - State " + outputState
+ "</p>");
            // If the outputState is off, it displays the ON
button
            if (outputState=="off") {
              client.println("<p><a  href=\"/26/on\"><button
class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/26/off\"><button
class=\"button button2\">OFF</button></a></p>");
            }

            // The HTTP response ends with another blank line
            client.println();
            // Break out of the while loop
            break;
            } else { // if you got a newline, then clear
currentLine
            currentLine = "";
            }
      } else if (c != '\r') {  // if you got anything else
but a carriage return character,
            currentLine += c;       // add it to the end of
the currentLine
      }
      }
      }
      // Clear the header variable
      header = "";
      // Close the connection
      client.stop();
      Serial.println("Client disconnected.");
      Serial.println("");
  }
}
```

4. Lalu buka **Serial Monitor** dan amati IP *address* yang tertera pada Serial Monitor.

Gambar 2. 7 IP address pada serial monitor

5. Kemudian buka *browser*, lalu masukkan IP *address* pada poin 3. Tunggu hingga halaman *website* ditampilkan dengan baik. Pastikan perangkat yang mengakses IP *address* ESP32 harus berada pada jaringan yang sama, sebagai contoh: ESP32

terkoneksi dengan *router* yang sama dengan PC praktikan, ESP32 terkoneksi dengan *tethering hotspot* dari ponsel praktikan.



Gambar 2. 8 Halaman Website ESP32

6. Tekan tombol ON/OFF pada halaman *website*, lalu amati hasil yang terjadi pada rangkaian praktikan.

7. Lakukan modifikasi berupa penambahan jumlah LED menjadi 2 buah pada rangkaian dan modifikasi kodenya sehingga *website* dapat mengendalikan tambahan LED tersebut

2.7 PERTANYAAN ANALISIS

1. Jelaskan mekanisme pengaturan pin output sedemikian hingga mengeluarkan nyala lampu sesuai dengan yang diinginkan. Lengkapi dengan penjelasan register yang digunakan!
2. Jelaskan bagaimana pengaturan pin input! Apa pengaruh resistor pull up?
3. Bagaimana proses pengaturan timer sebagai pengganti fungsi delay? Sertakan perhitungannya.
4. Bagaimana proses pengaturan interrupt dengan menggunakan timer ? Sertakan perhitungannya.
5. Bagaimana proses pengaturan interupsi tombol pada ESP32? Sertakan register yang terlibat dan fungsinya.
6. Jelaskan proses request dan response pada protokol HTTP secara sederhana!

2.8 DAFTAR DATA PRAKTIKUM

1. Percobaan 2.5.1 Output Digital: Tabel Perilaku LED terhadap Waktu
2. Percobaan 2.5.2 Input Digital: Tabel Perilaku LED terhadap Waktu dan Penekanan Tombol (Resistor Eksternal)
3. Percobaan 2.5.2 Input Digital: Grafik Tegangan Pin I/O terhadap Waktu saat Tombol Ditekan (Resistor Eksternal)
4. Percobaan 2.5.2 Input Digital: Tabel Perilaku LED terhadap Waktu dan Penekanan Tombol (Resistor Internal)
5. Percobaan 2.5.2 Input Digital: Grafik Tegangan Pin I/O terhadap Waktu saat Tombol Ditekan (Resistor Internal)
6. Percobaan 2.5.3 Timer: Tabel Perilaku LED terhadap Waktu (Sesuai Gambar 2.3 Kiri)
7. Percobaan 2.5.3 Timer: Tabel Perilaku LED terhadap Waktu (Sesuai Gambar 2.3 Kanan)
8. Percobaan 2.5.4 Interupsi Timer: Tabel Perilaku LED terhadap Waktu (Pilih Salah Satu Pola)
9. Percobaan 2.5.5 Interupsi Tombol: Tabel Perilaku LED terhadap Waktu dan Penekanan Tombol (Rising Edge)
10. Percobaan 2.5.6 Wi-Fi dan IoT: Tabel Perilaku LED terhadap Penekanan Tombol pada Web Server
11. Percobaan 2.6.1 Output Digital: Tabel Perilaku LED terhadap Waktu
12. Percobaan 2.6.2 Input Digital: Tabel Perilaku LED terhadap Waktu dan Penekanan Tombol (Resistor Internal)
13. Percobaan 2.6.2 Input Digital: Grafik Tegangan Pin I/O terhadap Waktu saat Tombol Ditekan (Resistor Internal)
14. Percobaan 2.6.3 Timer: Tabel Perilaku LED terhadap Waktu (Pilih Salah Satu Pola pada Gambar 2.3)
15. Percobaan 2.6.4 Interupsi Timer: Tabel Perilaku LED terhadap Waktu (Pilih Salah Satu Pola)
16. Percobaan 2.6.5 Interupsi Tombol: Tabel Perilaku LED terhadap Waktu dan Penekanan Tombol (Rising Edge)
17. Percobaan 2.6.6 Wi-Fi dan IoT: Tabel Perilaku LED terhadap Penekanan Tombol pada Web Server


INSTITUT TEKNOLOGI BANDUNG
1920

MODUL 3

ADC, DAC, DAN KOMUNIKASI

3.1 TUJUAN

- Praktikan mampu membuat aplikasi ADC pada Arduino Nano
- Praktikan mampu membuat aplikasi DAC dengan menggunakan DAC yang dimiliki oleh ESP32
- Praktikan mampu menggunakan Level Converter sesuai dengan kondisi mikrokontroler dan modul sensor yang digunakan
- Praktikan mampu membuat aplikasi komunikasi Serial antara 2 Arduino Nano
- Praktikan mampu membuat aplikasi komunikasi dengan menggunakan SPI antara 2 arduino Nano
- Praktikan mampu membuat aplikasi komunikasi dengan menggunakan I2C pada hubungan arduino nano dan modul I2C

3.2 LANDASAN TEORI

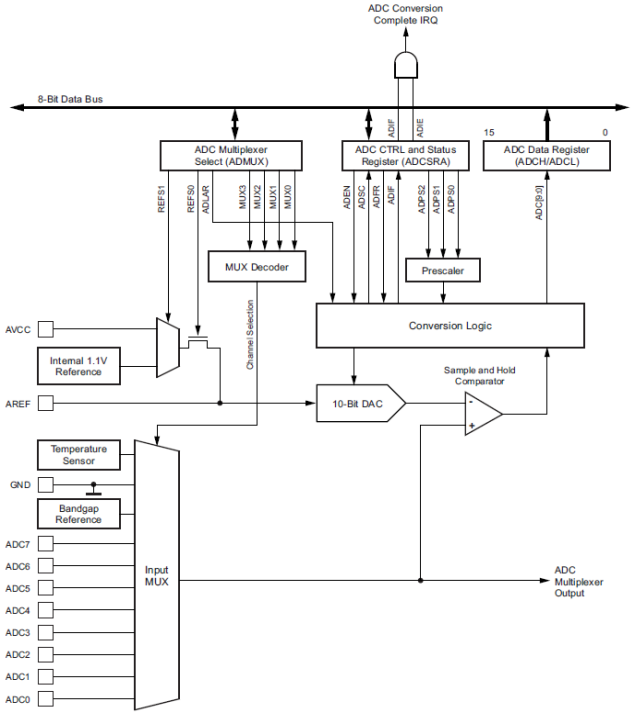
3.2.1 ADC

ADC (Analog to Digital Converter) merupakan sebuah sistem yang mengubah sinyal analog menjadi sinyal digital yang telah terkuantisasi. Setiap level tegangan akan dipetakan pada level-level biner/digital tertentu. Sinyal analog perlu diubah menjadi sinyal digital untuk memungkinkan pemrosesan data secara digital pada mikrokontroler. Sebagian besar mikrokontroler memiliki ADC.

ADC bekerja dengan memetakan tegangan masukan ke level tegangan terkuantisasi. ADC memiliki resolusi dalam memetakan level tegangan tersebut. Pada ADC 10 bit maka level tegangan referensi tertinggi akan memiliki nilai 1111111111 atau 1023 sedangkan level tegangan referensi terendah (biasanya 0V) akan memiliki nilai 0000000000 atau 0.

Arduino nano menggunakan mikrokontroler ATMega328p. Mikrokontroler ini memiliki ADC dengan resolusi 10 bit. ADC pada ATMega328p diatur pada beberapa register misalnya register ADMUX - ADC Multiplexer Selection Register dan register ADCSRA - ADC Control and Status Register A. [1]

Figure 23-1. Analog to Digital Converter Block Schematic Operation



Gambar 3. 1 Analog to Digital Converter Block Schematic Operation (Datasheet hal. 206)

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Register ADMUX

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Register ADCSRA

Berbeda dengan ATMega328, ESP32 memiliki ADC dengan resolusi 12 bit. ADC ESP 32 memiliki 4096 level tegangan yang berada pada rentang 0-4095 dan rentang tegangan 0-3.3V.

Konversi tegangan ke level kuantisasi ADC

$$\frac{V_{max}}{V_{in}} = \frac{Level\ kuantisasi\ maksimum}{output\ ADC}$$

$$Output\ ADC = \frac{Level\ kuantisasi\ maksimum\ .\ V_{in}}{V_{max}}$$

Keterangan :
Vmax = tegangan input maksimum
Vin = tegangan input

Contoh:
Diketahui input tegangan maksimum ESP32 adalah 3.3 V dengan resolusi ADC 12-bit. Tentukan nilai pembacaan ADC bila diberikan tegangan input 2V.

1. Menentukan level kuantisasi maksimum
   Resolusi ADC 12 bit berarti ESP32 memiliki 212 = 4096 level kuantisasi dengan rentang 0 - 4095. Oleh karena itu, level kuantisasi maksimumnya adalah 4095

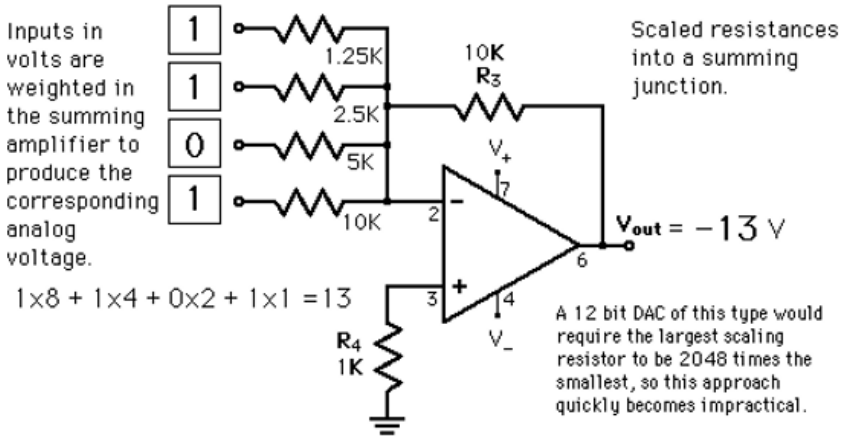
2. Menentukan pembacaan ADC
   Dari persamaan diatas didapatkan nilai pembacaan ADC adalah sebesar:

   $$\frac{2}{3.3} * 4095 = 2481.8181$$

   Karena output ADC berupa integer, maka nilai yang terbaca pada ADC adalah 2481.

3.2.2 DAC

Digital to Analog Converter (DAC) merupakan sebuah sistem yang berfungsi untuk mengubah nilai biner/digital menjadi nilai analog. Secara umum terdapat beberapa metode implementasi DAC, salah satunya dengan menggunakan opamp yang dioperasikan sebagai summing amplifier. [2]

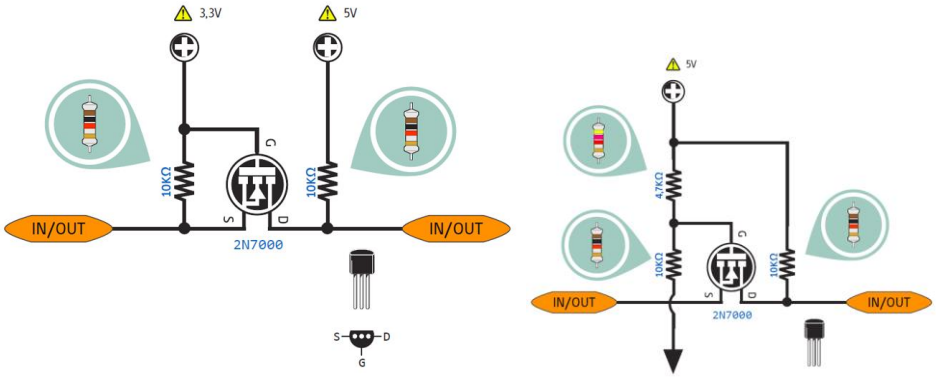


Gambar 3. 2 Rangkaian 4 bit DAC dengan summing amplifier

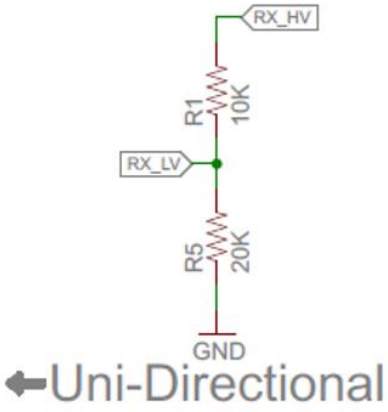
Gambar diatas merupakan rangkaian summing amplifier untuk 4 bit DAC. DAC dibutuhkan bila sebuah mikrokontroler berhubungan dengan perangkat yang membutuhkan sinyal analog sebagai masukannya. Secara umum, output dari DAC akan berbentuk sinyal dengan level tegangan tertentu yang bersifat diskrit. Untuk mendapatkan sinyal yang lebih halus dapat ditambahkan kapasitor sebagai filter rekonstruksi (bersifat lowpass).

3.2.3 Rangkaian Level Converter

Rangkaian level converter berfungsi untuk mengubah/menyesuaikan nilai level tegangan logika pada mikrokontroler. Beberapa mikrokontroler memiliki level tegangan yang berbeda sehingga diperlukan level converter agar keduanya dapat berfungsi sesuai dengan spesifikasi kerjanya masing-masing. Secara umum, terdapat dua jenis level converter yaitu bi-directional level converter dan Uni-directional level converter. Sesuai dengan namanya, bi-directional level converter dapat digunakan untuk dua arah konversi antara dua level tegangan sedangkan uni-directional level converter hanya dapat digunakan untuk satu arah saja. Berikut merupakan contoh rangkaian dari level converter antara 3.3 V dan 5 V. [3]



Gambar 3. 3 Bi-directional Level Converter 3.3V - 5V

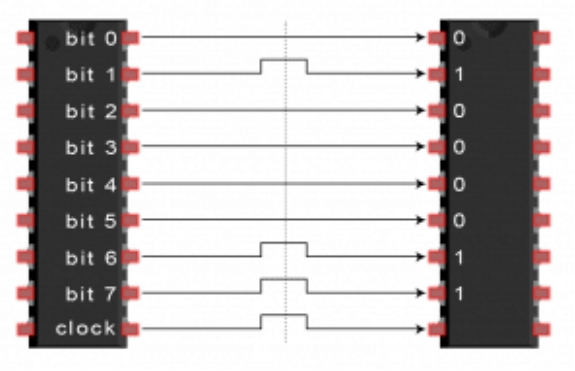


Gambar 3. 4 Uni-directional Level Converter

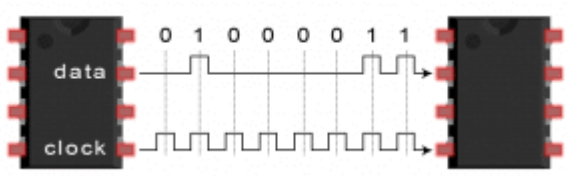
Dua gambar di atas merupakan contoh rangkaian level converter. Bi-directional level converter menggunakan transistor dan resistor sedangkan uni-directional level converter dapat diimplementasikan secara sederhana dengan menggunakan resistor yang berfungsi sebagai pembagi tegangan.

3.2.4 I2C dan SPI

I2C (Inter Integrated Circuit) dan SPI (Serial Peripheral Interface) merupakan salah satu contoh komunikasi serial antar mikrokontroler. Komunikasi serial adalah komunikasi dimana bit data dikirimkan satu persatu melalui sebuah kabel. Berbeda dengan komunikasi paralel dimana seluruh bit data dikirimkan secara paralel dengan menggunakan lebih dari satu kabel.

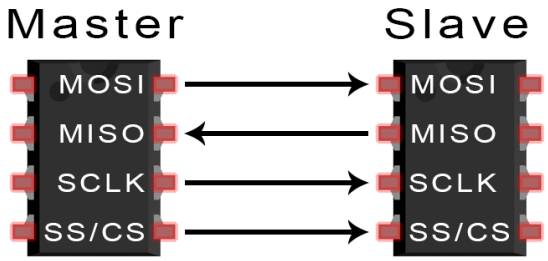


Gambar 3. 5 Komunikasi Paralel



Gambar 3. 6 Komunikasi Serial

SPI adalah protokol komunikasi yang banyak digunakan oleh perangkat-perangkat elektronik seperti modul SD Card, RFID card reader module serta 2.4 GHz wireless transmitter/receivers. Salah satu kelebihan dari SPI adalah pengiriman data tanpa interupsi. Sejumlah bit data dapat dikirimkan/diterima pada sebuah stream data yang kontinyu. Pada I2C maupun UART, data dikirimkan dalam bentuk paket dan terbatas pada beberapa jumlah bit. Pada protokol tersebut juga diperlukan start/stop bit yang menandakan titik mulai/berhenti masing-masing paket sehingga terdapat interupsi pada saat transmisi. [5]



Gambar 3. 7 komunikasi SPI

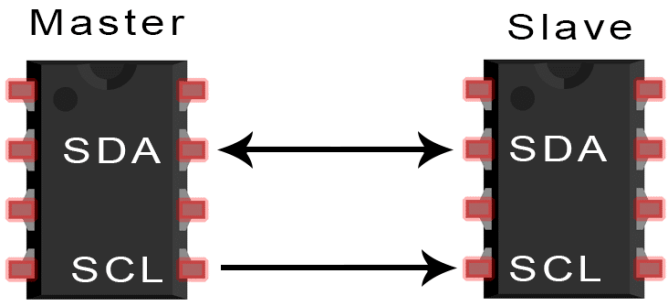
Keterangan :

MOSI (Master Output/Slave Input)	:	Untuk mengirim data dari master
MISO (Master Input/Slave Output)	:	Untuk mengirim data dari slave
SCLK (Clock)	:	Clock
SS/CS (Slave Select/Chip Select)	:	Kabel untuk memilih slave

Pada komunikasi SPI terdapat hubungan master dan slave. Perangkat yang berperan sebagai master mengendalikan clock pada komunikasi. Biasanya perangkat master berupa mikrokontroler sedangkan slave biasanya berupa

sensor, display maupun chip memori yang menerima data/instruksi dari master.

Beberapa kelebihan dari komunikasi SPI adalah tidak adanya interupsi pada pengiriman data akibat start/stop bit, pemisahan MISO dan MOSI memungkinkan pengiriman data dari master dan slave secara bersamaan serta kecepatan transfer data yang cenderung lebih tinggi dari I2C.



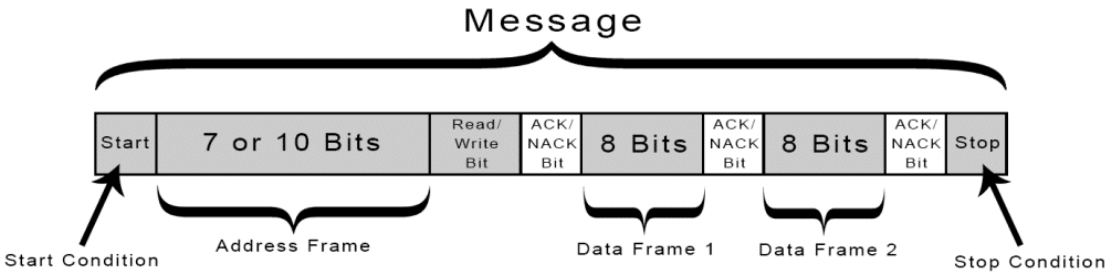
Gambar 3. 8 Komunikasi I2C

Keterangan:

SDA : Serial Data

SCL : Serial Clock

Komunikasi I2C memiliki fitur yang mirip dengan SPI namun pengiriman datanya mirip dengan komunikasi UART. Komunikasi I2C juga terdapat master dan slave. Seperti komunikasi SPI, pada komunikasi I2C juga memungkinkan komunikasi dengan slave yang jumlahnya lebih dari satu. Komunikasi I2C menggunakan 2 kabel saja alih-alih menggunakan 4 kabel seperti pada komunikasi SPI. Komunikasi I2C juga merupakan komunikasi serial yang bersifat sinkron (Tersinkronisasi pada clock). [6]



Gambar 3. 9 Struktur pengiriman data pada komunikasi I2C

Gambar di atas merupakan struktur pengiriman data pada komunikasi I2C. Seperti komunikasi UART, terdapat bit start dan stop yang menandakan dimulai dan diakhirinya pengiriman data. Pada komunikasi I2C terdapat address frame yang berisi alamat dari slave yang ingin dituju. Komunikasi ini memungkinkan komunikasi dengan slave yang lebih dari satu yang masing-masing memiliki alamat yang unik.

Beberapa kelebihan dari komunikasi I2C adalah penggunaan kabelnya yang lebih sedikit yaitu 2 buah saja, memungkinkan komunikasi dengan slave yang lebih dari satu. Komunikasi I2C merupakan salah satu protokol komunikasi yang banyak digunakan

Referensi

[1]. http://www.hdhprojects.nl/2017/12/07/atmega328-with-adc/, diakses pada 1 Februari 2021 pukul 12.35

[2]. https://components101.com/articles/digital-to-analog-converters-dac, diakses pada 2 Februari 2021 pukul 14.16

[3]. Slide kuliah EL3014-digital-input-2019-02-15

[4]. Slide kuliah EL3014 - 40 - komunikasi serial

[5]. https://www.circuitbasics.com/basics-of-the-spi-communication-protocol, diakses pada 2 Februari 2021 pukul 14.56

[6]. https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/, diakses pada 2 Februari 2021 pukul 15.14

3.3 TUGAS PENDAHULUAN

1. Membuat semua program (source code) yang diperlukan untuk masing-masing percobaan (sertakan keterangan-keterangan penting pada source code menggunakan komentar); Jelaskan masing-masing baris atau bagian kode tersebut.

3.4 ALAT DAN KOMPONEN YANG DIGUNAKAN

- Arduino Nano (2 buah)
- PCF8754P (1 buah)
- Level Converter MH (1 buah)
- Resistor beberapa jenis (secukupnya)
- Multimeter (1 buah)
- Button (2 buah)
- LED (1 buah)

3.5 . ADC

Persiapan / Setting awal

- Sebelum Arduino Nano dihubungkan pada daya listrik, buatlah rangkaiannya terlebih dahulu.
- Hubungkan pin Vin Arduino Nano pada rangkaian voltage divider.
- Input sinyal ADC adalah beberapa level tegangan analog yang diperoleh dari voltage divider dengan mengganti nilai nilai resistornya.
- Gunakan 2 buah resistor untuk rangkaian voltage divider dengan R1 bernilai tetap dan R2 divariasikan.

3.5.1 Membaca Nilai Sinyal Analog

1. Gunakan R1 dan R2 masing masing bernilai 2k.
2. Lengkapi kode dibawah kemudian jalankan program tersebut.

Tabel 3. 1 Kode Membaca Nilai Sinyal Analog

```
#define pinAnalog A0

int sinyalADC;

void setup() {
  //Pilih nilai port serial
  //Pendefinisian mode pinAnalog
}

void loop() {
  //Baca nilai sinyal analog
  //Print nilai sinyal analog
}
```

3.5.2 Mengubah ke Nilai Tegangan yang Terukur

1. Modifikasi program sebelumnya untuk mengkonversikan nilai sinyal analog yang terbaca ke nilai tegangan yang terukur di Arduino IDE. Kemudian buat flowchart program yang telah dimodifikasi tersebut.
2. Catat nilai sinyal analog dan nilai tegangan yang terukur untuk beberapa level tegangan dengan R1 = 2k (tetap) dan R2 = 390, 1k, 1.2k, 1.5k, 2k, 2.2k, 2.7k, 3.3k, 3.9k.
3. Buat grafik perbandingan antara nilai tegangan sebenarnya (secara teori dan menggunakan multimeter) dengan nilai tegangan yang terukur di ADC pada Microsoft Excel.

3.6 . DAC

Persiapan / Setting awal

- Sebelum Arduino Nano dihubungkan pada daya listrik, buatlah rangkaiannya terlebih dahulu.

- Hubungkan pin GPIO25 ESP32 pada pin A0 Arduino Nano.

3.6.1 Pembuatan Sinyal Analog Sinusoidal Resolusi 9-bit dengan Metode Lookup Table

Langkah-langkah pembuatan Lookup Table sinyal sinusoidal 1khz dengan resolusi 9-bit untuk DAC dengan resolusi 9 bit menggunakan Ms Excel.

1. Buat sinyal fasa digital 9 bit 1 siklus (i = 0 - 511).
2. Standardisasi sinyal fasa yang dibuat. (nilai = 0 - 1)
3. Hitung amplitudo sinyal sinusoidal dari sinyal fasa tersebut.
4. Ubah amplitudo sinyal sinusoidal menjadi bentuk sinyal digital 8-bit (0-255)
5. Hitung perkiraan sinyal voltase keluaran DAC

3.6.2 Simulasi Pengiriman dan Pembacaan Sinyal menggunakan DAC ESP32

1. Buka Source Code yang digunakan untuk percobaan DAC dengan Arduino IDE.
2. Jalankan contoh program dibawah ini pada arduino pertama untuk mengirimkan sinyal digital pada DAC. Isi parameter yang diperlukan pada contoh program!

Tabel 3. 2 Kode Simulasi Pengiriman dan Pembacaan Sinyal menggunakan DAC

```
#include <Wire.h>
#include <math.h>

//Membuat sinusoid 1kHz dengan look up table 9 bit
#define ResTable 512

//Definisikan Frekuensi Sinusoid
#define freq 1000

// DAC
#define DAC1 25

int i;
int DACInput;

float delayVal = //Masukan nilai delayVal agar frekuensi
yang diterima nano receiver benar-benar 1kHz ;

const PROGMEM uint16_t DAC_LookUpTable[512] = {

// isi dengan sinyal digital sinusoidal dalam format 8-bit

};

void setup(void){
  Serial.begin(9600);
}
```

```
void loop(void){
  for(i = 0; i < ResTable ; i++){
      DACInput = pgm_read_word(&(DAC_LookUpTable[i]));
      dacWrite(DAC1, DACInput);
     delayMicroseconds(round(delayVal*1000000));
  }
}
```

Hint:

Untuk mempermudah penginputan sinyal digital 8 bit dapat dilakukan hal berikut :

- Copy sinyal dari Ms Excel dan di paste text only pada Ms Word
- Replace ^p with ,

Besarnya delay yang diperlukan dapat dihitung dari resolusi dan frekuensi sinyal sinusoidal yang digunakan.

3. Jalankan contoh program dibawah ini pada arduino kedua untuk membaca sinyal analog keluaran DAC. Lakukan juga pengamatan dengan menggunakan osiloskop.

Tabel 3. 3 Kode membaca sinyal analog keluaran DAC

```
//ADC internal nano adalah 10 bit, nilai maksimal 1024
#define ResADC 1023
#define Vin 4.68 //Voltage Nano di sekitar situ
#define analogPin A0
int value;
float voltage;


void setup() {
  Serial.begin(9600);
}

void loop() {
  value = analogRead(analogPin);
  voltage = (float)value/ResADC*Vin;
  Serial.println(voltage,3);
}
```

Catat hasil percobaan yang didapatkan, analisis hasil tersebut, dan buat flowchart program yang digunakan

3.7 . KOMUNIKASI SERIAL

Persiapan / Setting awal

- Sebelum Arduino Nano dihubungkan pada daya listrik, buatlah rangkaiannya terlebih dahulu.
- Hubungkan pin Tx Arduino Nano 1 (Master) ke pin Rx Arduino Nano 2 (Slave) serta pin Rx Arduino Nano 1 ke pin Tx Arduino Nano 2.

3.7.1 Mengirimkan dan Menampilkan Data dari Arduino Nano 1 (Master) ke Arduino Nano 2 (Slave) Melalui Port Serial

1. Lengkapi kode dibawah kemudian jalankan program tersebut.

Tabel 3. 4 Kode untuk Master

```
char dataMaster1 = '1';
char dataMaster2 = '0';

void setup() {
      //Pilih nilai port serial
}

void loop() {
      //Tulis dataMaster1 ke port serial
      //Berikan delay
      //Tulis dataMaster2 ke port serial
      //Berikan delay
}
```

Tabel 3. 5 Kode untuk Slave

```
char dataSlave;

void setup() {
  //Pilih nilai port serial
}

void loop() {
  //Baca nilai pada port serial dan definisikan sebagai
dataSlave
  //Cetak dataSlave
 }
```

2. Amati keluaran pada TX pada Arduino Master dengan osiloskop. Analisis sinyal yang dikeluarkan TX.

3.7.2 Mengendalikan kondisi LED Menggunakan Button Melalui Port Serial

1. Hubungkan button pada Arduino Nano 1 (Master) dan LED pada Arduino Nano 2 (Slave).
2. Kemudian modifikasi program sebelumnya sehingga LED pada Arduino Nano 2 dapat dikendalikan kondisinya oleh button pada Arduino Nano 1.
3. Amati dan catat hasil percobaan diatas serta buatlah flowchartnya.

3.8 . LEVEL CONVERTER MH

Persiapan / Setting awal

- Kebutuhan komponen :
  - Arduino Nano (1 buah)
  - Level Converter MH (1 buah)
  - Breadboard dan kabel-kabel
  - ESP32 (1 buah)
  - LCD I2C (1 buah)
- Arduino IDE

3.8.1 Melakukan Konversi Tegangan dengan Level Converter MH

1. Hubungkan Arduino Nano dengan Level Converter MH.
   Prinsip menghubungkan :
   - Level Converter memiliki dua sisi, HV (High Voltage) dan LV (Low Voltage). Pin HV dan LV adalah tegangan referensi konversi tegangan pada masing-masing sisi.
   - Untuk mengubah logic tegangan HV ke LV, input ke salah satu pin pada sisi HV dari Level Converter dan ambil output tegangan yang lebih rendah pada pin LV yang bersesuaian.
   - Untuk mengkonversi logic tegangan LV ke HV, input ke pin LV dan output pada pin HV yang bersesuaian.

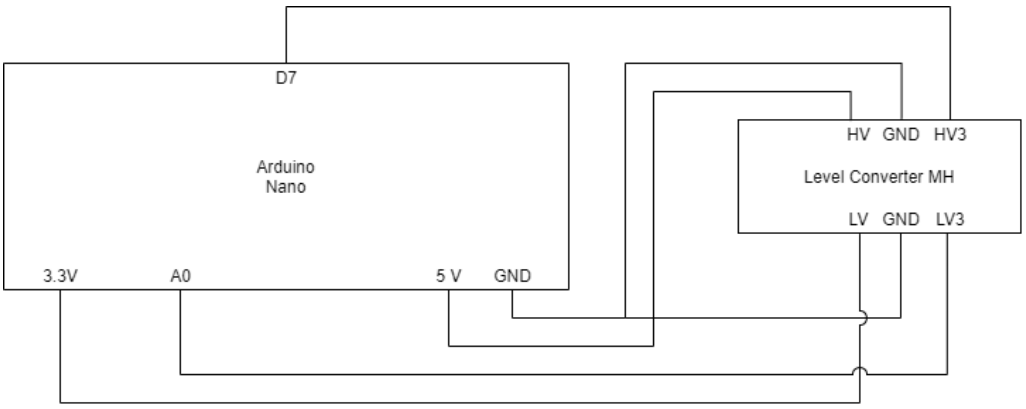
   Catatan :
   Hubungan input-output konversi pada Level Converter terjadi pada pin HV dan LV yang bersesuaian (memiliki angka yang sama dan berseberangan).

2. Pertama, set nilai tegangan referensi pada sisi pin HV dan LV.
   Untuk pin HV, beri input tegangan 5 V dari Arduino.
   Untuk pin LV, beri input tegangan 3.3 V dari Arduino.
   Jangan lupa untuk menghubungkan pin GND.

3. Arduino memiliki tegangan logic pada 5 V. Hubungkan salah satu pin digital Arduino dengan salah satu pin pada sisi HV Level Converter. Pada tutorial ini, digunakan pin D7 pada Arduino dan pin HV3 pada Level Converter.

4. Ambil output dari Level Converter (pada kasus ini, pin LV3) kemudian baca nilainya pada Arduino dengan menginputkannya pada pin analog pada Arduino. (Pada tutorial ini, digunakan pin A0) pada Arduino.

5. Skematik



Gambar 3. 10 Skematik Level Converter MH

6. Kode Arduino

Pada bagian atas dari program, masukkan kode berikut untuk deklarasi pin dan konstanta lainnya.

Tabel 3. 6 Kode Arduino untuk deklarasi

```
//ADC internal nano adalah 10 bit, nilai maksimal 1024
#define ResADC 1023
#define Vin 4.68 // Voltage Nano di sekitar situ

// pin yang digunakan
#define digitalOutput 7
#define analogPin A0
```

Pada bagian void setup(), inisialisasi serial communication dengan Serial.begin() dan set pin yang digunakan sebagai INPUT/OUTPUT dengan fungsi pinMode().

Pada bagian program utama di void loop(), munculkan tegangan yang dibaca hasil konversi pada serial monitor. Set agar nilai tegangan yang dikonversi berubah antara LOW dan HIGH setiap 1 detik.

Struktur dari kode pada void loop ini dapat dilihat sebagai berikut.

Tabel 3. 7 Kode pada void loop

```
void loop() {
  /* Konversi tegangan HIGH */
  // set pin menjadi HIGH
  // baca nilai input tegangan dari output level converter
  // kirim melalui serial communication
  delay(1000);

  /* Konversi tegangan LOW */
  // set pin menjadi LOW
  // baca nilai input tegangan dari output level converter
```

```
  // kirim melalui serial communication
  delay(1000);
}
```

Keterangan :
Silakan membuat struktur program sendiri, yang penting fungsionalitasnya sama.

7. Gunakan osiloskop untuk mengamati sinyal pada HV dan LV pada level converter.
8. Amati output yang dihasilkan pada Serial Monitor dan plot dengan Serial Plotter.
   Hasil yang diharapkan : Saat logic HIGH, level tegangan turun ke 3.3 V. Saat logic LOW, level tegangan tetap 0 V.

3.8.2 Interfacing ke Tegangan yang Lebih Tinggi dengan Level Converter MH

Saat ini hanya tersedia mikrokontroler ESP32 (logic HIGH di 3.3 V) dan diinginkan membuat tampilan antarmuka menggunakan LCD I2C (logic HIGH di 5 V).

Kemampuan ESP32 untuk mengeluarkan nilai tegangan yang lebih rendah ini mungkin tidak dapat memberikan catu daya untuk menyalakan LCD I2C tersebut.

Selain itu, lebih rendahnya nilai tegangan logic dari ESP32 (ditambah dengan gangguan noise dan drop tegangan) mungkin saja membuat LCD I2C tidak dapat membaca data yang dikirim dengan benar sehingga komunikasinya (terutama komunikasi I2C) tidak reliable.

Langkah Percobaan :

1. Menggunakan Level Converter MH, lakukan interfacing antara ESP32 dengan LCD I2C agar ESP32 dapat menampilkan data dengan baik pada LCD I2C.
2. Amati sinyal pada pin SCL dan SDA dengan osiloskop.
   Catatan :
   LCD I2C berkomunikasi dengan mikrokontroler ESP32 menggunakan I2C. Pastikan pin SCL dan SDA kedua komponen dapat berkomunikasi. Perhatikan juga kebutuhan catu daya LCD I2C.
   Library yang digunakan : LCD_I2C.h

3.9 . SPI

Persiapan / Setting awal

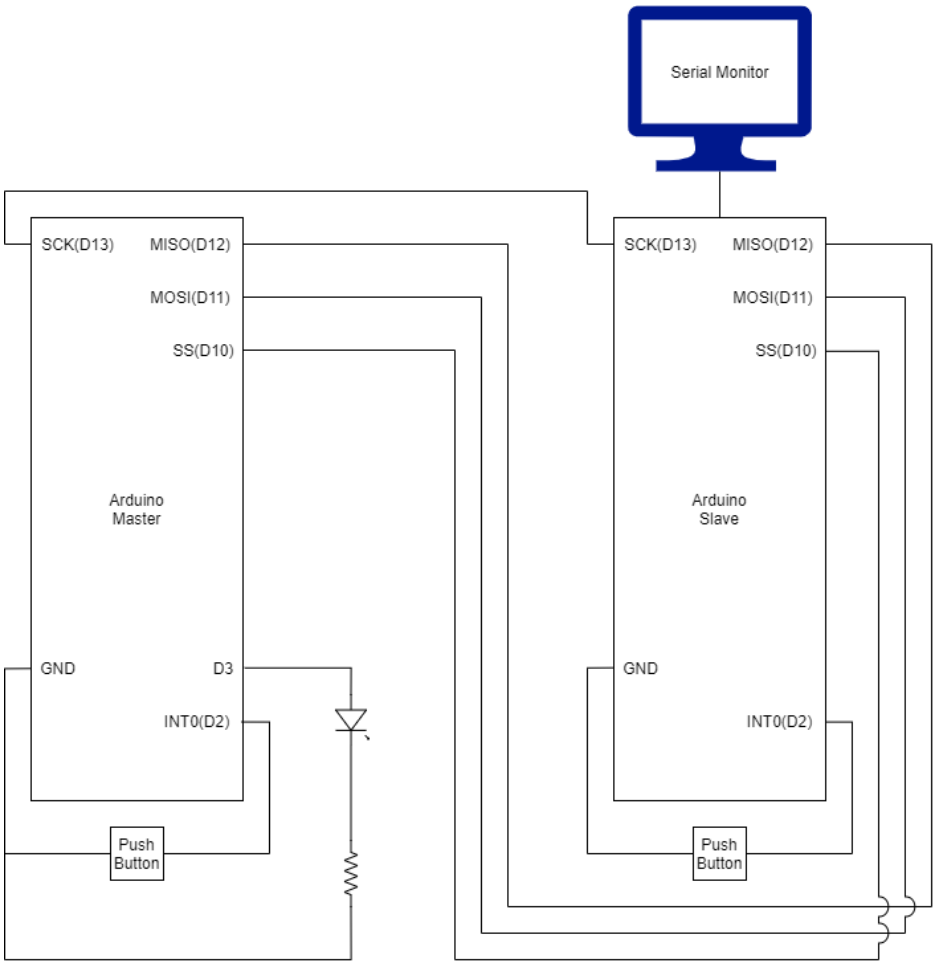
- Kebutuhan komponen :
  - Arduino Nano (2 buah)
  - LED (1 buah)
  - Push Button (2 buah)

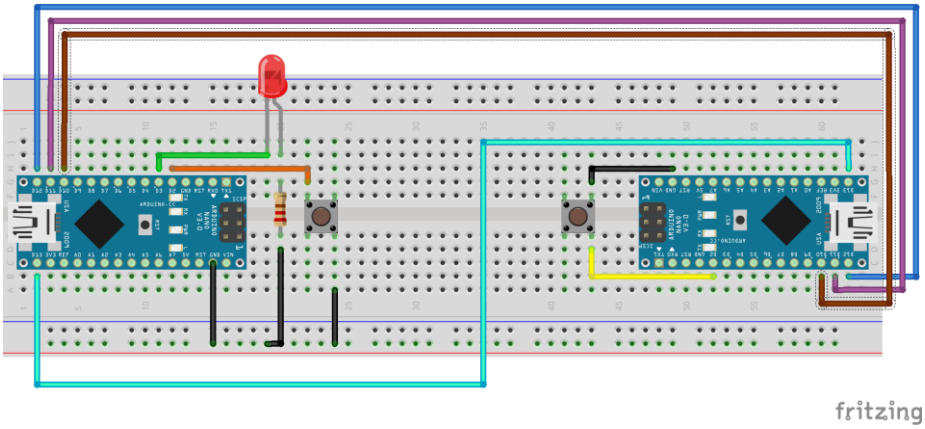
- Resistor secukupnya
- Breadboard dan kabel-kabel

- Arduino IDE sudah terinstall dengan library SPI.h

3.9.1 Melakukan Komunikasi SPI 2 Arduino

1. Pastikan library SPI.h sudah terinstall pada Arduino IDE.
2. Hubungkan pin-pin komunikasi SPI dari kedua Arduino.
   Pin-pin SPI pada Arduino Nano: MISO (D12), MOSI(D11), SCK(D13), SS(D10).
3. Pada Arduino pertama (selanjutnya disebut sebagai Master), hubungkan komponen berikut.
   - LED pada salah satu pin digital (pada tutorial ini digunakan pin D3).
   - Push Button pada pin external interrupt (pada tutorial ini digunakan pin INT0 (D2)).
   - Resistor sesuai kebutuhan.
4. Pada Arduino kedua (selanjutnya disebut sebagai Slave), hubungkan komponen berikut.
   - Push Button pada pin external interrupt (pada tutorial ini digunakan pin INT0 (D2)).
   - Resistor sesuai kebutuhan.
5. Skematik





Gambar 3. 11 Skematik Komunikasi SPI 2 Arduino

6. Tugas : Saat button pada slave ditekan, LED yang terkoneksi pada server akan menyala. Saat button pada master ditekan, Slave akan merespon dengan mengirimkan data serial ke monitor berupa "Tombol Master ditekan!"
7. Kode Master
   Persiapan :

Tabel 3. 8 Kode Deklarasi variabel global, konstanta, pin

```
#include <SPI.h>
#define interruptPin 2 //INT0
#define LED 3

bool pushed = 0;
int dataSend;
int dataRx;
```

Tabel 3. 9 Kode Inisialisasi pin, serial, dan interrupt pada void setup.

```
pinMode(interruptPin, INPUT);
pinMode(LED,OUTPUT);
attachInterrupt(digitalPinToInterrupt(interruptPin),
SPITransmit, FALLING);
Serial.begin(9600);
```

Lakukan inisialisasi komunikasi SPI untuk master sebagai berikut pada void setup.

Tabel 3. 10 Kode inisialisasi komunikasi SPI untuk master

```
//SPI Init
//Lakukan proses inisiasi SPI
```

Pada void loop, masukkan kode berikut untuk melakukan komunikasi SPI dan menghandle LED.

Tabel 3. 11 Kode komunikasi SPI dan menghandle LED

```
// Memulai Komunikasi dengan Slave
//Mengirim data ke slave, dan di saat bersamaan, menerima
data dari slave. Data dari slave tersebut kemudian
dianalisis. Jika data bernilai 1,hidupkan LED selama 1
detik.
```

Jangan lupa membuat kode untuk melakukan interupsi tombol saat tombol ditekan.

Keterangan :
Silakan modifikasi apabila menggunakan pin yang berbeda.
Perhatikan dan cari peran dari hal-hal berikut.

- Line `SPI.setClockDivider(SPI_CLOCK_DIV8);`
- Fungsi `SPI.transfer`
- Pin SS

8. Kode Slave
   Persiapan : deklarasi pin, konstanta, dan variabel lainnya.

Tabel 3. 12 Kode deklarasi pin, konstanta, dan variabel lainnya

```
#include <SPI.h>
#define interruptPin 2 //INT0

bool pushed = 0;
int dataSend;
int dataRx;
```

Tabel 3. 13 Kode Inisialisasi pin, interrupt, dan serial komunikasi pada void setup.

```
pinMode(interruptPin, INPUT);
attachInterrupt(digitalPinToInterrupt(interruptPin),
ButtonPushed, FALLING);
Serial.begin(9600);
```

Lakukan inisiasi SPI, Suaojab interupsi SPI, dan jika slave menerima data 1 dari master, kirimkan sebuah pesan ke serial monitor berupa “Tombol master telah ditekan”. Jangan lupa membuat interupsi tombol.

Keterangan :
Silakan modifikasi apabila menggunakan pin yang berbeda.
Perhatikan dan cari peran dari hal-hal berikut.

- Line `SPCR |= _BV(SPE);`
- ISR(SPI_STC_vect)
- Register SPCR dan SPDR

9. Hubungkan Slave dengan Serial Monitor pada Arduino IDE.
10. Tekan push button pada Slave. Amati hal yang terjadi.
    Hasil yang diharapkan : LED pada Master menyala.
11. Tekan push button pada Master. Amati hal yang terjadi.
    Hasil yang diharapkan : Pesan berhasil dicetak pada Serial Monitor.

Debug :
Jika tidak bisa menggunakan internal pull up untuk push button, gunakan external pull up resistor (silakan modifikasi kode sesuai kebutuhan).

3.10 . I2C

Persiapan / Setting awal

- Kebutuhan komponen :
  - Arduino Nano (1 buah)
  - PCF8574 (1 buah)
  - LED (1 buah)
  - Resistor secukupnya
  - Breadboard dan kabel-kabel
- Arduino IDE sudah terinstall dengan library PCF8745.h

3.10.1 Menggunakan I2C untuk Komunikasi dengan Module I2C

Module I2C untuk LCD menggunakan konsep GPIO extender sehingga kebutuhan penggunaan 16 pin untuk menyalakan LCD dapat dilakukan dengan hanya dua pin yaitu SCL dan SDA dengan komunikasi I2C.

Pada percobaan ini, praktikan akan melakukan pengaksesan chip dari module tersebut yaitu PCF8574 sebagai GPIO extender (bukan keseluruhan module untuk menyalakan LCD yang akan dilakukan pada praktikum selanjutnya). Konsep GPIO extender ini akan dilihat dengan mengatur sebuah LED setelah melewati interfacing PCF8574.

Praktikan diharapkan dapat :

- Menggunakan library PCF8574.h
- Mengatur Address dari PCF8574
- Menulis Data di sebuah alamat register pada PCF

Langkah Percobaan :

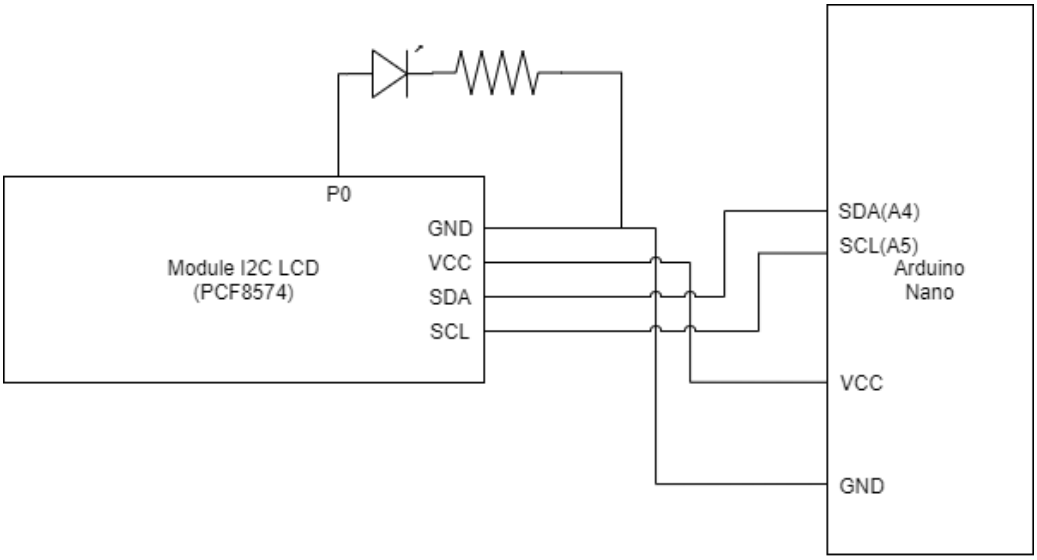
1. Pastikan library PCF8574.h sudah terinstall pada Arduino IDE. https://github.com/xreef/PCF8574_library
2. Hubungkan PCF8574 dengan Arduino Nano melalui pin I2C (SCL dan SDA) serta pin catu daya (VCC,GND)

3. Pasang LED pada salah satu pin digital dari PCF8574 (pada tutorial ini digunakan pin P0). Jangan lupa untuk menambahkan resistor
4. Amati dan atur address dari module I2C LCD PCF8574. Untuk melakukan hal tersebut, amati bagian A0, A1, A2 pada module.

Gambar 3. 12 Mengatur nilai tegangan dari pin A0, A1, dan A2

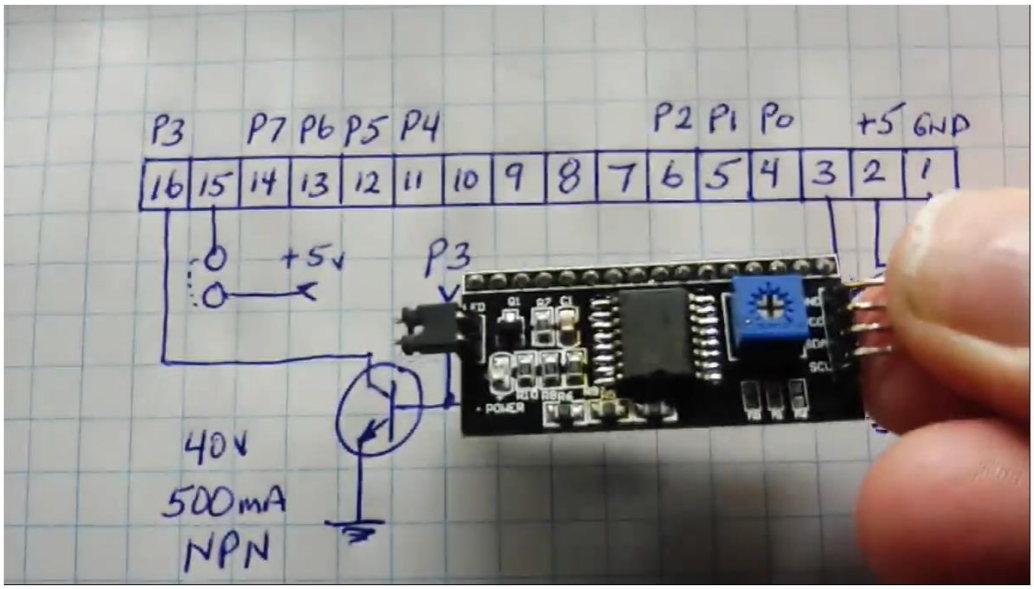
Selanjutnya, sesuai dengan manufacturer chip dan konfigurasi pin A0, A1, dan A2, alamat dari module ini dapat bervariasi.

5. Skematik.



Gambar 3. 13 Skematik Komunikasi dengan Module I2C

Untuk lebih jelasnya dari pin-pin pada module I2C LCD (selain keempat pin GND, VCC, SDA, SCL), perhatikan gambar berikut.



Gambar 3. 14 Visualisasi cara menghubungkan PCF8574 dengan Arduino Nano

6. Mulai buat program Arduino untuk mengedipkan LED pada PCF8574 setiap 1 detik (berubah setiap 1 detik).
   Persiapan kode :

   - Untuk menggunakan PCF8574, include library PCF8574 terlebih dahulu.

Tabel 3. 14 Include library PCF8574

```
#include <PCF8574.h>
```

   - Untuk melakukan pengalamatan PCF8574, deklarasi objek dengan class PCF8574 dengan input argumen alamat PCF8574 pada constructor.

Tabel 3. 15 Deklarasi objek dengan class PCF8574

```
PCF8574 pcf8574(/* alamat PCF8574 */);
```

   Selanjutnya, tutorial ini akan didemonstrasikan dengan objek pcf8574 dengan class PCF8574.

7. Inisialisasi pada void setup() :
   - Lakukan pengesetan pin-pin PCF8574 sebagai input/output dilakukan dengan method .pinMode() dengan argumen seperti saat pengesetan pin GPIO Arduino biasa.

Tabel 3. 16 Pengaturan Pin PCF

```
pcf8574.pinMode(/* PIN */,INPUT); // untuk INPUT
pcf8574.pinMode(/* PIN */,OUTPUT); // untuk OUTPUT
```

Catatan : addressing pin pada PCF8574 dilakukan dengan prefix P dilanjutkan dengan angka (P0, P1, P2, …, P7), cukup set pin yang digunakan saja (seperti pada kode Arduino biasa)

- Inisialisasi PCF8574 dengan memanggil method .begin() pada objek PCF8574 yang dibuat.

Tabel 3. 17 Inisialisasi PCF8574

```
pcf8574.begin();
```

8. Program utama pada void loop() :
   - Masukkan kode untuk mengedipkan LED yang dipasang pada PCF8574 setiap 1 detik.
   - Untuk mengeset nilai dari pin OUTPUT pada pcf8574, gunakan method .digitalWrite dengan argumen seperti pada fungsi digitalWrite biasa.

Tabel 3. 18 Program utama pada void loop

```
pcf8574.digitalWrite(/* PIN */, HIGH); // untuk HIGH
pcf8574.digitalWrite(/* PIN */, LOW); // untuk LOW
```

9. Upload kode ke Arduino dan amati hal yang terjadi.
   Hasil yang diharapkan : LED pada PCF8574 berubah state setiap 1 detik.

3.11 PERTANYAAN ANALISIS

1. Jelaskan proses konversi hasil ADC ke nilai sebenarnya pada arduino Nano!
2. Jelaskan proses pembuatan sinyal sinusoid dengan menggunakan Look Up Table! Apa keuntungan penggunaan Look-Up Table?
3. Jelaskan struktur data dari protokol yang digunakan dalam komunikasi serial!
4. Mengapa diperlukan level converter?
5. Jelaskan kegunaan pin-pin yang terlibat dalam komunikasi SPI! Sertakan juga bagaimana prosedur komunikasi menggunakan SPI (Termasuk struktur datanya)!
6. Jelaskan kegunaan pin-pin yang terlibat dalam komunikasi I2C! Sertakan juga bagaimana prosedur komunikasi menggunakan I2C (Termasuk struktur datanya)!
7. Sebutkan keuntungan dan kerugian menggunakan SPI dan I2C!

3.12 DAFTAR DATA PRAKTIKUM

1. Percobaan 3.5.2 Mengubah ke Nilai Tegangan yang Terukur: Tabel Nilai Level ADC Terbaca dan Nilai Tegangan Terkait
2. Percobaan 3.6.2 Simulasi Pengiriman dan Pembacaan Sinyal menggunakan DAC ESP32: Grafik Tegangan terhadap Waktu pada Serial Plotter Arduino IDE

3. Percobaan 3.6.2 Simulasi Pengiriman dan Pembacaan Sinyal menggunakan DAC ESP32: Grafik Tegangan terhadap Waktu pada Osiloskop
4. Percobaan 3.7.1 Mengirimkan dan Menampilkan Data dari Arduino Nano 1 (Master) ke Arduino Nano 2 (Slave) Melalui Port Serial: Data yang Dikirimkan oleh Arduino Master dan Data yang Diterima oleh Arduino Slave
5. Percobaan 3.7.1 Mengirimkan dan Menampilkan Data dari Arduino Nano 1 (Master) ke Arduino Nano 2 (Slave) Melalui Port Serial: Grafik Data pada Kanal TX dari Arduino Master pada Osiloskop (untuk 1 byte data)
6. Percobaan 3.7.2 Mengendalikan kondisi LED Menggunakan Button Melalui Port Serial: Tabel Perilaku LED terhadap Tombol
7. Percobaan 3.8.1 Melakukan Konversi Tegangan dengan Level Converter MH: Grafik Data Input dan Output dari Logic-Level Converter MH untuk pin SCL dan pin SDA pada Osiloskop
8. Percobaan 3.8.2 Interfacing ke Tegangan yang Lebih Tinggi dengan Level Converter MH: Grafik Data Input dan Output dari Logic-Level Converter MH untuk pin SCL dan pin SDA pada Osiloskop
9. Percobaan 3.9.1 Melakukan Komunikasi SPI 2 Arduino: Tabel Perilaku LED terhadap Tombol
10. Percobaan 3.9.1 Melakukan Komunikasi SPI 2 Arduino: Grafik Data MISO dan MOSI pada Osiloskop
11. Percobaan 3.10.1 Menggunakan I2C untuk Komunikasi dengan Module I2C: Tabel Perilaku LCD I2C terhadap Waktu
12. Percobaan 3.10.1 Menggunakan I2C untuk Komunikasi dengan Module I2C: Grafik Data SDA dan SCL pada Osiloskop


INSTITUT TEKNOLOGI BANDUNG
1920

MODUL 4

OPERATING SYSTEM

4.1 TUJUAN

- Praktikan memahami penggunaan MPU6050 yang memanfaatkan protokol komunikasi I2C
- Praktikan mampu membuat produk sederhana dengan memanfaatkan Operating system

4.2 LANDASAN TEORI

4.2.1 Real- Time Operating System (RTOS)

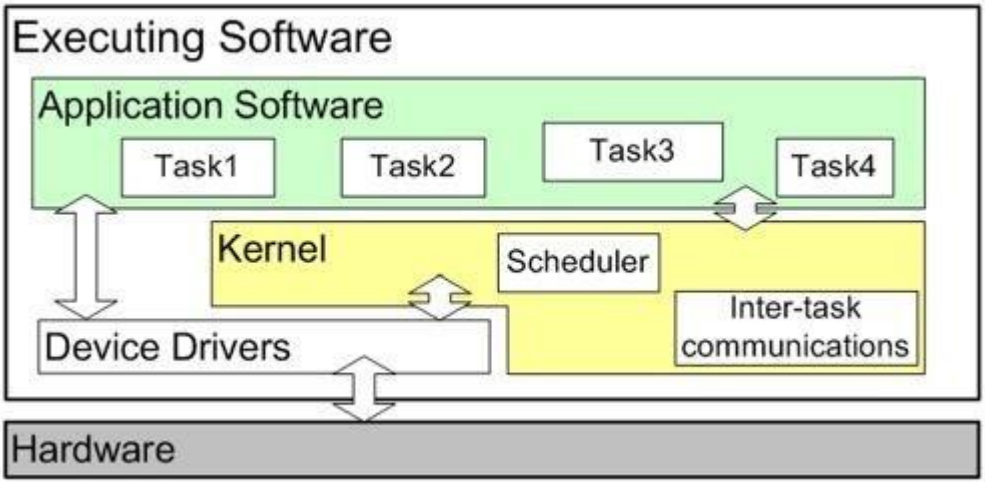
Real- Time Operating System (biasa di sebut RTOS) adalah sebuah Operating System (OS) yang digunakan untuk memenuhi kebutuhan aplikasi secara Real Time pada Embedded Device yang memproses data secara langsung tanpa ada nya penundaan (Buffer). Real Time karena system ini hamper bekerja setiap saat dimana ia dibutuhkan saat itu juga. Salah satu kelebihan Operating System RTOS adalah kemampuan nya untuk melakukan kerja secara konsisten baik secara waktu yang ia butuhkan maupun secara task aplikasi yang mampu ia kerjakan.

RTOS dibutuhkan karena pada system Embedded karena biasa nya pada system Embedded , digunakan sebuah mikrokontroler dengan prosesor tunggal (Single Processor), sehingga pada pekerjaan Embedded System yang membutuhkan system secara Real Time dan melakukan lebih dari satu pekerjaan, dibutuhkan sebuah Operating System yang dapat melakukan penjadwalan (Scheduling) beberapa pekerjaan sehingga dapat dilakukan dalam sebuah prosesor tunggal, dan mudah dimodifikasi untuk melakukan berbagai pekerjaan. Karakter dasar dari Operating System RTOS adalah sebuah sistem yang mempunyai beberapa konsekuensi yang akan berpengaruh pada sistem apabila deadline (batas akhir waktu pelaksanaan sebuah pekerjaan) tidak terpenuhi.

RTOS sendiri terdiri dari 2 jenis yaitu, sistem soft RTOS dan sistem hard RTOS. Soft RTOS bisa dideskripsikan sebagai sistem yang hampir selalu menyelesaikan task dengan waktu yang telah ditentukan. Pada soft RTOS kemungkinan penyelesaian task melewati batas waktu pelaksanaan task masih bisa terjadi. Dan pada sistem soft RTOS, apabila terjadi kegagalan mencapai

deadline dalam waktu yang telah ditentukan maka sistem akan mengalami efek yang tidak begitu berbahaya bagi sistem. Contohnya seperti penurunan performa sistem. Sedangkan hard RTOS merupakan system yang dipastikan selalu menyelesaikan task dalam waktu yang telah ditentukan. Dikatakan pasti selalu menyelesaikan task karena hard RTOS selalu menyelesaikan task sebelum deadline dan apabila terjadi kegagalan menyelesaikan task maka sistem akan mengalami efek berbahaya yang dapat merusak sistem secara keseluruhan.

Diagram arsitektur cara kerja RTOS dapat dilihat sebagai berikut.



Gambar 4. 1 Arsitektur Cara Kerja *Real Time Operating System* (RTOS)

Terdapat 2 komponen utama dalam RTOS, yaitu Tugas (atau *Task)* dan Kernel (atau *Scheduler)*.

Tugas (Task)

Tugas (atau biasa di sebut Task) adalah sebuah objek/program yang dapat dieksekusi dan beranggapan mempunyai CPU untuk task itu sendiri. Salah satu proses perancangan aplikasi dengan RTOS yaitu membagi semua pekerjaan dalam aplikasi tersebut menjadi beberapa bagian task. Tiap task merupakan loop yang akan terus berulang. Dalam proses pengulangan tersebut, task akan mengalami tiga buah keadaan yaitu: (i) Running, merupakan keadaan di mana sebuah task dengan prioritas tertinggi berjalan, (ii) Ready, merupakan keadaan yang dialami sebuah task jika terdapat sebuah task lain sedang running dan task yang berada pada State ready akan melanjutkan pengerjaan task yang sempat tertunda oleh task yang lebih tinggi prioritasnya. (iii) Blocked, merupakan keadaan di mana jika sebuah task membutuhkan event atau data maka akan masuk ke dalam blocked hingga event atau data yang dibutuhkan telah tersedia.

Gambar 4. 2 *State Diagram* dari State *Task* pada RTOS

Kernel

Kernel merupakan salah satu bagian dari sistem multitasking yang mempunyai fungsi sebagai manajemen dari seluruh task, mengatur komunikasi tiap task dan yang terpenting adalah mengatur pewaktuan (Clock) untuk CPU sehingga tidak terjadi tabrakan (Crash) Task pada CPU. Terdapat 2 jenis Kernel :

- Non- Preempetive Kernel

  Non-preemptive scheduling biasa dikenal dengan nama lain cooperative multitasking, di mana task bekerja sama satu sama lain untuk berbagi CPU. ISR bias membuat sebuah task dengan prioritas tertinggi menjadi siap untuk dieksekusi, tetapi kemudian ISR akan kembali ke task yang sebelumnya mendapat interupsi. Task yang sudah siap tadi akan berjalan apabila task yang mendapat interupsi tadi sudah selesai berjalan atau dengan kata lain task yang sudah selesai berjalan akan menyerahkan CPU kepada task dengan prioritas tertinggi.

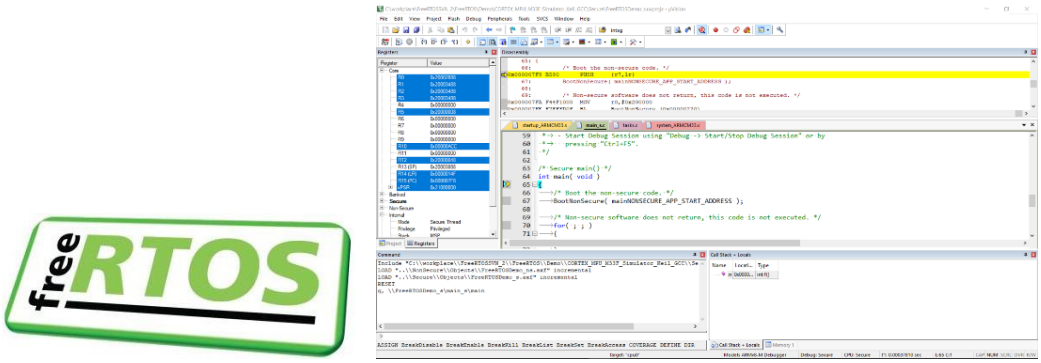
- Preempetive Kernel

  Preemptive kernel banyak digunakan untuk membuat aplikasi dengan RTOS. Hal ini karena preemptive kernel mempunyai respons yang lebih bagus daripada non-preemptive kernel. Dari gambar 2.3 dapat dijelaskan prinsip kerja dari preemptive kernel. Di mana task dengan prioritas tertinggi yang sudah siap dieksekusi akan langsung berjalan. Dan jika pada saat itu sedang ada task dengan prioritas yang lebih rendah berjalan maka task dengan prioritas rendah tersebut akan ditunda. Jadi dapat

disimpulkan bahwa preemptive kernel selalu mendahulukan task dengan prioritas tertinggi yang siap untuk dieksekusi.

Clock tick merupakan interupsi special yang muncul secara periodik. Clock tick bias dianggap sebagai detak jantung dari sistem yang berfungsi sebagai dasar untuk menentukan timer pada sistem Real Time dengan RTOS. Waktu untuk tiap munculnya clock tick bisa ditentukan pada saat merancang sistem RTOS. Selain itu dalam penggunaan RTOS terdapat istilah **Semaphore**. Semaphore dalam sistem RTOS adalah penanda yang menandakan kapan suatu Task dapat dilakukan dan kapan suatu Task tidak dapat dilakukan.

Terdapat beberapa jenis RTOS, seperti SafeRTOS dan FreeRTOS. Jenis Operating System RTOS yang paling sering digunakan adalah **FreeRTOS**, karena jenis RTOS yang bersifat Open Source, gratis dan mudah digunakan.



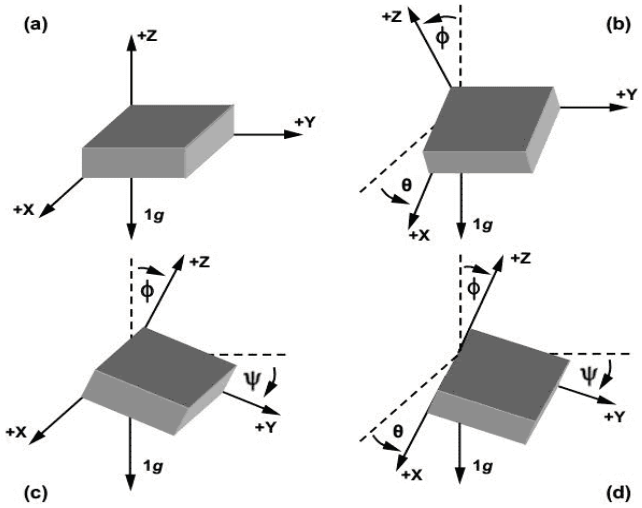
Gambar 4. 3 Lambang Aplikasi dan Penampilan Aplikasi *Operating System* RTOS FreeRTOS

4.2.2 MPU6050

Sensor MPU6050 adalah sensor yang mampu membaca kemiringan sudut berdasarkan data dari sensor Accelerometer dan sensor Gyroscop. Sensor ini juga dilengkapi oleh sensor suhu yang dapat digunakan untuk mengukur suhu di keadaan sekitar. Jalur data yang digunakan pada sensor ini adalah jalur data I2C dan mampu berjalan pada tegangan sumber sebesar 3 sampai 5 V. Penampilan Sensor Accelerometer dan Gyroscope MPU6050 dapat dilihat sebagai berikut.

Gambar 4. 4 Penampilan Sensor MPU6050

Sensor MPU6050 memiliki sensor Keseimbangan (Gyroscope) dan sensor kecepatan (Accelerometer) bersama dengan sensor suhu. Modul ini berukuran sangat kecil, memiliki konsumsi daya yang rendah, sangat akurat, toleransi guncangan yang tinggi, dan memiliki harga yang murah. Selain itu Sensor MPU6050 juga memiliki modul bawaan yang dapat digunakan dalam berbagai mikrokontroler yang sering digunakan, seperti Arduino dan ESP32. MPU6050 dapat dengan mudah dihubungkan dengan sensor lain seperti Magnetometer . Giroskop yang ada di Sensor MPU6050 dapat mendeteksi rotasi pada tiga sumbu yaitu sumbu- x , sumbu - y, dan sumbu- Z. Efek Coriolis menyebabkan getaran saat Gyros diputar di sekitar sumbu mana pun. Getaran ini ditangkap oleh kapasitor, yang kemudian sinyal yang dihasilkan kemudian diperkuat, didemodulasi dan di- Filter untuk menghasilkan tegangan yang sebanding dengan kecepatan sudut. Tegangan ini kemudian didigitalisasi menggunakan modul Analog to Digital Converter (ADC). Pergerakan pada sumbu- x , sumbu- y dan sumbu- z yang dapat diukur oleh sensor MPU6050 dapat dilihat sebagai berikut.

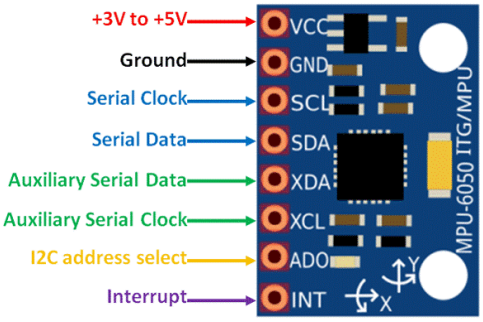


Gambar 4. 5 Pergerakan Pada Sumbu- x, Sumbu- y dan Sumbu - z yang Dapat Diukur oleh Sensor MPU6050

Spesifikasi umum dari sensor MPU6050 dapat dilihat sebagai berikut.

Chip model	MPU-6050
Power supply	3–5 V
Communication protocol	I2C
Gyroscope range	±250, ±500, ±1000, ±2000°/s
Accelerometer range	±2, ±4, ±8, ±16g
16-bit AD converter/16-bit data output	

Diagram Pin Input dan Output dari sensor MPU6050 dapat dilihat sebagai berikut. Pada sensor MPU6050, terdapat 8 buah pin, yaitu sebuah pin sumber tegangan (VCC), pin Ground , pin Serial Clock (SCL) , pin Serial Data (SDA), pin Auxillary Serial Data (XDA), pin Auxillary Serial Clock (XCL), pin I2C Address Select (AD0), dan pin interupsi (INT). Diagram pin MPU6050 dan keterangan detail nya dapat dilihat sebagai berikut.



Gambar 4. 6 Diagram Pin *Input* dan Pin *Output* Sensor MPU6050

Tabel 4. 1 Tabel Keterangan Detail Pin *Input* dan *Output* Sensor MPU6050

MPU6050 Pinout		
Pin#	**Pin Name**	**Description**
01	Vcc	This pin used for Supply Voltage. Its input voltage is +3 to +5V.
02	GND	This pin use for ground
03	SCL	This pin is used for clock pulse for I2C compunction
04	SDA	This pin is used for transferring of data through I2C communication.
05	Auxiliary Serial Data (XDA)	It can be used for other interfaced other I2C module with MPU6050.
06	Auxiliary Serial Clock (XCL)	It can also be used for other interfaced other I2C module with MPU6050.
07	AD0	If more than one MPU6050 is used a single MCU, then this pin can be used to vary the address.
08	interrupt (int)	This pin is used to indicate that data is available for MCU to read.

Referensi

[1] Leksono, Bayu Puji et al. PENERAPAN REAL TIME OPERATING SYSTEMS (RTOS) PADA MIKROKONTROLER AVR (STUDI KASUS CHIBIOS/RT) . Universitas Diponegoro (UNDIP) : Makalah Seminar Tugas Akhir. 2016.
[2]https://aninditablog.wordpress.com/2012/05/14/free-real-time-operating-systems-freertos/, Diakses pada 3 Februari 2021
[3] Kharisma, Oktaf B. et al. Implementasi Sensor MPU 6050 untuk Mengukur Kesetimbangan Self Balancing Robot Menggunakan Kontrol PID . Seminar Nasional Teknologi Informasi, Komunikasi dan Industri (SNTIKI-10).
[4] MPU6050 : Pin Diagram, Circuit Working, Specifications & Applications (elprocus.com), Diakses pada 3 Februari 2021
[5] Introduction to MPU6050 - The Engineering Projects , Diakses pada 3 Februari 2021.

4.3 TUGAS PENDAHULUAN

1. Membuat semua program (source code) yang diperlukan untuk masing-masing percobaan (sertakan keterangan-keterangan penting pada source code menggunakan komentar); Jelaskan masing-masing baris atau bagian kode tersebut.

4.4 ALAT DAN KOMPONEN YANG DIGUNAKAN

- ESP32 (1 buah)
- LED (1 buah)
- Sensor MPU6050 (1 buah)
- LCD(16x2) plus I2C (1 buah)
- Resistor dan Jumper (secukupnya)
- Busur (Pengukur Sudut)

4.5 . SENSOR MPU6050

4.5.1 Menggunakan Sensor MPU6050

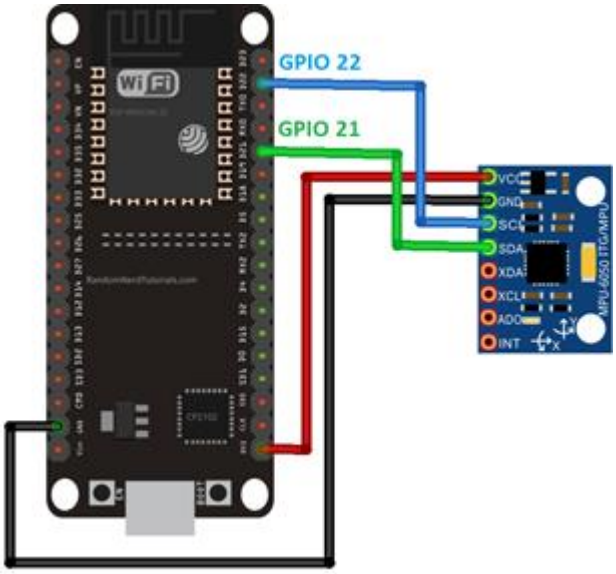
Langkah Percobaan :

1. Membuat rangkaian yang menghubungkan sensor MPU6050 dengan Board ESP32 pada *breadboard* dengan konfigurasi PIN sebagai berikut.

Tabel 4. 2 Konfigurasi PIN

MPU6050	ESP32
VCC	3.3V atau 5V

GND	GND
SCL	GPIO22
SDA	GPIO21



Gambar 4. 7 Skematik Rangkaian sensor MPU6050 dan Board ESP32

2. Membuka Arduino IDE. Pastikan Driver Board ESP32 telah terinstal sehingga dapat digunakan pada Arduino IDE.
3. Membuat sebuah program baru dengan mengklik file -> new. Simpan program tersebut dengan nama mpu6050.ino
4. Pada awal program, masukkan library wire.h untuk mengakses fitur I2C dengan menuliskan kode berikut.

Tabel 4. 3 Kode Memasukkan library

```
#include "Wire.h"
```

5. Selanjutnya definisikan alamat I2C dari sensor MPU6050 dengan nama MPU_ADDR (lihat *datasheet* dari sensor MPU6050 untuk memperoleh alamat I2C sensor).

Tabel 4. 4 Kode Mendifinisikan Alamat I2C

```
#define MPU_ADDR (..)
```

6. Mendeklarasikan variable-variable yang akan digunakan.

Tabel 4. 5 Kode Mendeklarasikan Variable

```
int16_t accX, accY, accZ;
float rangePerDigit = .000061f;
float NormAccX,NormAccY,NormAccZ;
int pitch, roll;
```

7. Pada bagian setup(), inisialisasi serial monitor dengan baudrate 9600.

Tabel 4. 6 Kode Inisialisasi Serial Monitor

```
Serial.begin(9600);
```

8. Masih pada bagian setup(), lakukan komunikasi I2C dengan sensor MPU6050 untuk membangunkan sensor. Sensor dapat dibangunkan dengan mengakses register PWR_MGMT_1 dan menulisnya dengan 0 (buka *datasheet* untuk memperoleh alamat dari register PWR_MGMT_1).

Tabel 4. 7 Kode Membangunkan Sensor

```
Wire.begin();
Wire.beginTransmission(..);
Wire.write(..); //Power Management untuk MPU6050
Wire.write(..); //Membangunkan MPU
Wire.endTransmission(true);
```

9. Pada bagian loop(), tulis kode untuk memulai transmisi untuk mengambil nilai akselerasi. Berdasarkan *datasheet* yang diberikan, nilai akselerasi untuk tiap sumbu disimpan pada dua buah register berukuran 8-bit. Misal, nilai akselerasi di sumbu-x disimpan pada register accX_H dan acc_X. accX_H menyimpan 8-bit msb dan accX_L menyimpan 8-bit lsb dari accX. Untuk mengakses semua nilai akselerasi tiap sumbu, tulis perintah untuk menulis alamat register accX_H sebagai alamat awal. Untuk mengakses register-register berikutnya, gunakan fungsi wire.requestFrom() dengan panjang 6 byte.

Tabel 4. 8 Kode Inisiasi Sensor

```
Wire.beginTransmission(..);
Wire.write(..); //Alamat Awal
Wire.endTransmission(false);   //Agar   transimisi   tetap
berjalan
Wire.requestFrom(.., .., true);
```

10. Berikutnya, baca respon yang dikirimkan sensor dengan memanggil fungsi Wire.read(). Fungsi ini akan me-*return* respon yang dikirim sensor secara urut dari alamat awal. Untuk menggabungkan 8-bit msb dan 8-bit lsb menjadi 16 bit lsb, dapat menggunakan operasi bitwise shift left dan OR. Contoh isi variable acc dan nilai yang di-*return- oleh* Wire.read() ditunjukkan oleh tabel berikut.

Tabel 4. 9 Variable acc dan nilai yang di-*return- oleh* Wire.read()

accX	Wire.read()
0	accX_H (8-bit)
accX_H (8-bit)	accX_L (8-bit)
accX (16-bit)	accY_H (8-bit)

Implementasi algoritma tersebut ke dalam *source code*!

Tabel 4. 10 Kode Implementasi Algoritma

```
//Pembacaan urut dari alamat accX
accX = ..;
accY = ..;
accZ = ..;
```

11. Nilai bacaan dari sensor perlu dinormaliasi untuk menghasilkan nilai akselerasi dalam satuan ms^{-2} . Tambahkan kode berikut ke dalam *source code*!

Tabel 4. 11 Kode Normalisasi

```
//Normalisasi Raw Data tersebut
NormAccX = accX * rangePerDigit * 9.80665f;
NormAccY = accY * rangePerDigit * 9.80665f;
NormAccZ = accZ * rangePerDigit * 9.80665f;
```

12. mengalkulasi *pitch* dan *roll* dengan menambahkan kode berikut ke dalam *source code*.

Tabel 4. 12 Kode Mengalkulasi *pitch* dan *roll*

```
//Mengkalkulasi pitch dan roll
pitch = -(atan2(NormAccX, sqrt(NormAccY*NormAccY +
NormAccZ* NormAccZ)) * 180.0) / M_PI;
roll = (atan2(NormAccY, NormAccZ)*180.0)/M_PI;
```

13. Menampilkan nilai *pitch* dan *roll* ke serial monitor dengan menambahkan kode berikut ke dalam *source code*.

Tabel 4. 13 Kode Menampilkan Nilai *pitch* dan *roll*

```
//Output Serial
  Serial.print("Pitch : ");
  Serial.print(pitch);
  Serial.print(" dan Roll : ");
  Serial.print(roll);
  Serial.println();
  delay(100);
```

14. Berikut adalah *Source code* lengkap dari langkah percobaan ini.

Tabel 4. 14 Kode Percobaan

```
#include "Wire.h"
#define MPU_ADDR (..)

int16_t accX, accY, accZ;
float rangePerDigit = .000061f;
float NormAccX,NormAccY,NormAccZ;
int pitch, roll;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Wire.begin();
  Wire.beginTransmission(..);
  Wire.write(..); //Power Management untuk MPU6050
  Wire.write(..); //Membangunkan MPU
  Wire.endTransmission(true);
}

void loop() {
  Wire.beginTransmission(..);
  Wire.write(..);
Wire.endTransmission(false);  //Agar transimisi tetap
berjalan
  Wire.requestFrom(.., .., true); //Akan mengakses 6
register

//Pembacaan urut dari alamat accX
  accX = ..;
```

```
  accY = ..;
  accZ = ..;

//Normalisasi Raw Data tersebut
  NormAccX = accX * rangePerDigit * 9.80665f;
  NormAccY = accY * rangePerDigit * 9.80665f;
  NormAccZ = accZ * rangePerDigit * 9.80665f;

  //Mengkalkulasi pitch dan roll
pitch   =   -(atan2(NormAccX,   sqrt(NormAccY*NormAccY   +
NormAccZ*NormAccZ))*180.0)/M_PI;
roll = (atan2(NormAccY, NormAccZ)*180.0)/M_PI;

  //Output Serial
  Serial.print("Pitch : ");
  Serial.print(pitch);
  Serial.print(" dan Roll : ");
  Serial.print(roll);
  Serial.println();
  delay(100);
}
```

15. *Compile* dan *upload* program ke *Board* ESP32.
16. Gerakkan *pitch* dan *roll* sensor MPU6050 dan amati nilainya pada serial monitor.
17. Bandingkan hasil yang muncul pada serial monitor dengan pengukuran langsung menggunakan busur

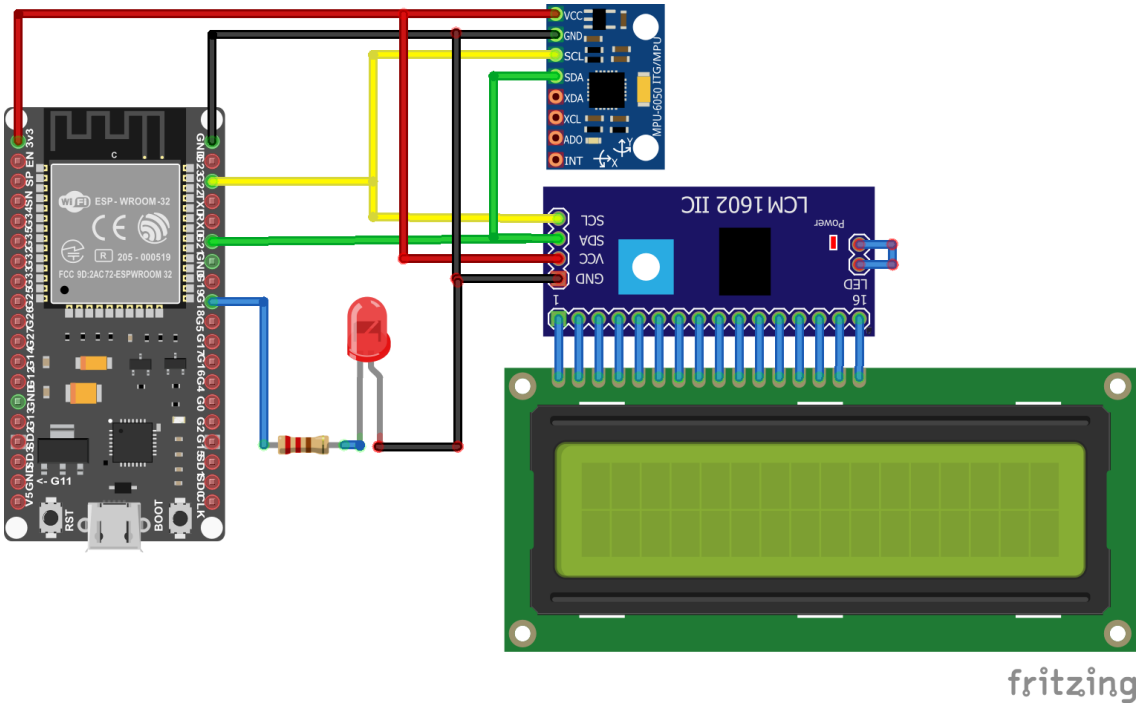
4.6 . RTOS

Persiapan / Setting Awal

- Menginstall Library FreeRTOS pada Arduino IDE

4.6.1 Menggunakan Sensor MPU6050 dengan RTOS

1. Modifikasi rangkaian pada percobaan IVA sehingga sesuai dengan skematik rangkaian berikut.



Gambar 4. 8 Skematik Rangkaian RTOS

2. Membuat *program* baru dengan nama rtos.ino.
3. Memasukkan *dependencies* dan definisi yang digunakan.

Tabel 4. 15 Kode Memasukkan *dependencies* dan definisi

```
#include "Wire.h"
#define MPU_ADDR (..)
#define rangePerDigit .000061f
#include <LiquidCrystal_I2C.h>
#define LED 19

//Tasks Priority
#define priorityTask1 2
#define priorityTask2 2
#define priorityTask3 2
```

4. Mendeklarasikan variabel-variabel dan objek yang digunakan.

Tabel 4. 16 Kode Mendeklarasikan Variabel dan Objek

```
int pitch, roll;
//Mutex Definition
SemaphoreHandle_t xMutex;


//Task Display
int lcdColumns = 16;
int lcdRows = 2;
//Membuat objek LCD
```

```
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);
```

5. Mendeklarasikan tiga buah fungsi yang akan dijalankan oleh task/thread.

Tabel 4. 17 Kode Mendeklarasikan Fungsi

```
//Sensing Task
void SensingTask(void *pvParam);

//LED Task
void BlinkTask(void *pvParam);

//Display Task
void DisplayTask(void *pvParam);
```

6. Pada bagian setup(), lengkapi potongan kode berikut.

Tabel 4. 18 Kode Setup

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  //Inisiasi LED
  pinMode(LED, OUTPUT);

  //Inisiasi MPU
  Wire.begin();
  Wire.beginTransmission(..);
  Wire.write(..); //Power Management untuk MPU6050
  Wire.write(..); //Membangunkan MPU
  Wire.endTransmission(true);

  //Inisiasi LCD
  lcd.init();
  lcd.backlight();

  //Mutex
  xMutex = xSemaphoreCreateMutex();

  //Task Start
  xTaskCreatePinnedToCore(..  ,  "Task  1"  ,  ..  ,  NULL,  ..,
NULL,..);
  xTaskCreatePinnedToCore(..  ,  "Task  2"  ,  ..  ,  NULL,  ..,
NULL,..);
  xTaskCreatePinnedToCore(..  ,  "Task  3"  ,  ..  ,  NULL,  ..,
NULL,..);
}
```

Pada bagian Task Start, buatlah tiga buah task/thread dengan parameter sebagai berikut.

Tabel 4. 19 Parameter Task Start

	Task Function	stackDepth	Priority	xCoreID
Task 1	SensingTask	2048	priorityTask1	0
Task 2	BlinkTask	2048	priorityTask2	1
Task 3	DisplayTasj	2048	priorityTask3	0

7. Implementasi dari fungsi SensingTask dengan melengkapi potongan kode berikut. Fungsi ini memiliki periode 100 ms. Gunakan macro pdMS_TO_TICKS untuk mengubah satuan milisekon menjadi ticks.

Tabel 4. 20 Kode Implementasi dari fungsi SensingTask

```
void SensingTask(void *pvParam) {
  (void) pvParam;
  int16_t accX, accY, accZ;
  float NormAccX, NormAccY, NormAccZ;
  while (1) {
      xSemaphoreTake(xMutex, portMAX_DELAY);
      {

      Wire.beginTransmission(..);
      //Memulai address register awal yang akan diakses di
MPU
      Wire.write(..);
      Wire.endTransmission(false); //Agar transimisi tetap
berjalan
      Wire.requestFrom(.., ..,  true); //Akan mengakses 6
register,

      //Pembacaan urut dari alamat awal
      accX = .. //Menggabungkan 2 register pertama menjadi
16 bit
      accY = .. //Menggabungkan 2 register kedua menjadi 16
bit
      accZ = ..//Menggabungkan 2 register ketiga menjadi 16
bit
      }
      xSemaphoreGive(xMutex);

      //Normalisasi Raw Data tersebut
      NormAccX = accX * rangePerDigit * 9.80665f;
      NormAccY = accY * rangePerDigit * 9.80665f;
      NormAccZ = accZ * rangePerDigit * 9.80665f;
      //Mengkalkulasi pitch dan roll
      pitch = -(atan2(NormAccX, sqrt(NormAccY * NormAccY +
NormAccZ * NormAccZ)) * 180.0) / M_PI;
      roll = (atan2(NormAccY, NormAccZ) * 180.0) / M_PI;
```

```
      vTaskDelay(..);
  }
}
```

8. Membuat implementasi dari fungsi BlinkTask dengan melengkapi potongan kode berikut. Lampu akan berkedip dengan interval 500 ms. Gunakan Gunakan macro pdMS_TO_TICKS untuk mengubah satuan milisekon menjadi ticks.

Tabel 4. 21 Kode Membuat Implementasi dari Fungsi BlinkTask

```
void BlinkTask(void *pvParam) {
  (void) pvParam;
  bool high = 0;
  while (1) {
      /*Impementasi kode agar LED berkedip jika bacaan
     Sensor melebihi threshold roll pada ±90 derajat
      dan mati jika bacaan sensor berada di rentang
      -90 sampai 90 derajat
      */

      vTaskDelay(..);
  }
}
```

9. Membuat implementasi dari fungsi DisplayTask dengan melengkapi potongan kode berikut. LCD akan refresh setiap 1 detik.

Tabel 4. 22 Kode Membuat Implementasi dari Fungsi DisplayTask

```
void DisplayTask(void *pvParam) {
  (void) pvParam;
  while (1) {
      xSemaphoreTake(xMutex, portMAX_DELAY);
      {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Pitch : ");
      lcd.print(pitch);
      lcd.setCursor(0, 1);
      lcd.print("Roll : ");
      lcd.print(roll);
      }
      xSemaphoreGive(xMutex);

      vTaskDelay(..);
  }
}
```

10. Biarkan isi fungsi loop() kosong.

Tabel 4. 23 Kode Fungsi Loop

```
void loop() {
}
```

11. *Compile* dan *upload* program ke *Board* ESP32.

    Gerakkan sensor MPU dan amati nilai *pitch* dan *roll* yang ditampilkan pada LCD. Amati juga apakah LED telah menyala sesuai nilai *roll* dari sensor.

4.7 PERTANYAAN ANALISIS

1. Jelaskan proses pembacaan data mentah MPU6050 dengan menggunakan I2C!
2. Jelaskan proses konversi data mentah MPU6050 menjadi data pitch dan roll!
3. Mengapa diperlukan Operating System?
4. Jelaskan apa kegunaan Mutex secara sederhana!
5. Buatlah flowchart dari produk yang telah kalian buat!

4.8 DAFTAR DATA PRAKTIKUM

1. Percobaan 4.5.1 Menggunakan Sensor MPU6050: Tabel Perbandingan Sudut Terukur oleh MPU6050 dan Busur Derajat
2. Percobaan 4.6.1 Menggunakan Sensor MPU6050 dengan RTOS: Tabel Perilaku LED terhadap Sudut Terbaca dari MPU6050
3. Percobaan 4.6.1 Menggunakan Sensor MPU6050 dengan RTOS: Tabel Penjadwalan Pelaksanaan State untuk Masing-Masing Task


INSTITUT TEKNOLOGI BANDUNG
1920

MODUL 5

PENGENDALI MOTOR DC

5.1 TUJUAN

- Praktikan dapat membuat antar muka mikrokontroler ke rotary encoder
- Praktikan dapat membaca informasi posisi dan kecepatan dari rotary encoder
- Praktikan dapat membuat antar muka output mikrokontroler ke motor DC
- Praktikan mampu membuat pengendali PID digital untuk mengendalikan posisi dan kecepatan motor DC

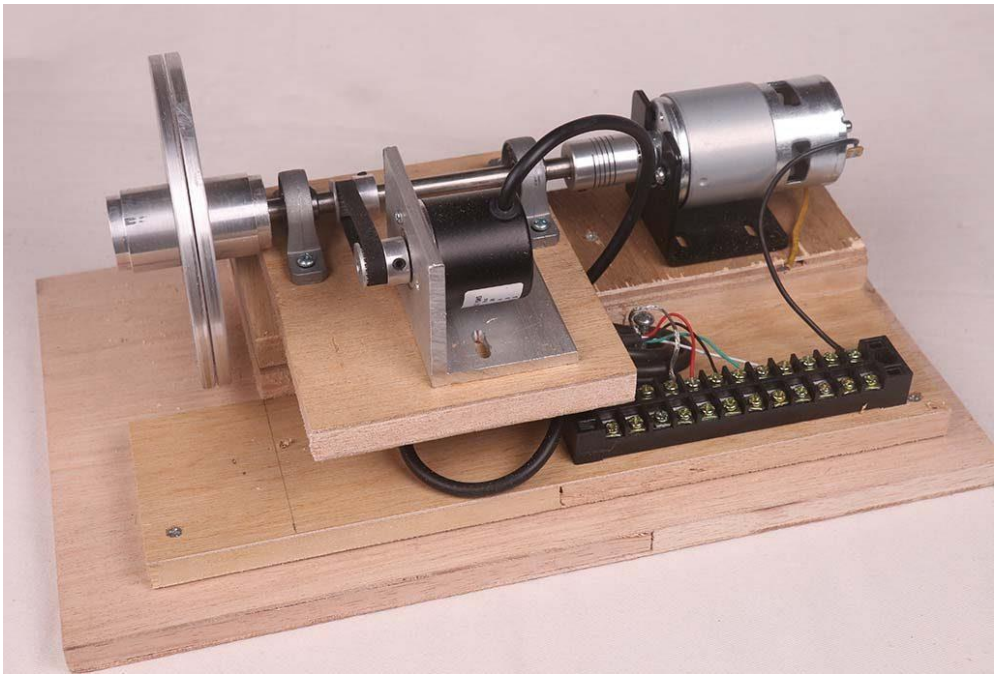
5.2 KOMPONEN YANG DIGUNAKAN

Berikut ini daftar komponen dan perangkat yang diperlukan untuk percobaan ini:

- Kit Motor DC
- Mikroprosesor Arduino Nano (ATMega328P)
- Kabel USB mini untuk Arduino Nano
- Breadboard
- Modul Motor Driver BTS7960
- Potensiometer dengan nilai 1k sampai 10k untuk input posisi
- Power Supply 5 volt 2 ampere untuk supply ke motor
- Osiloskop
- Multimeter
- Kabel-kabel pendek untuk breadboard

Kit Kendali Motor DC

Praktikum ini menggunakan kit motor DC sebagai berikut.



Gambar 5. 1 Kit Motor DC

Komponen utama kit motor DC adalah sebagai berikut.

- Motor DC 12 volt tipe 775 dengan poros 5 mm sebagai penggerak sistem.
- Poros utama stainless steel 8 mm
- Pillow bearing 8mm
- Rotary Encoder 400 pulse dengan poros 6 mm
- Dua buah Pelat Aluminium 5 mm sebagai beban momen inersia
- Closed loop timing belt dengan panjang 160 mm
- Pulley GT2 dengan poros 8 mm (untuk dipasang di poros utama)
- Pulley GT2 dengan poros 6 mm (untuk rotary encoder)
- Terminal blok 12 titik
  Flexible shaft coupling 8mm ke 5 mm

Gambar 5. 2 Pillow Bearing

Motor DC berputar sesuai dengan tegangan yang diberikan pada terminalnya. Arah putaran sesuai dengan polaritas tegangan. Motor DC menggerakkan poros

utama, yang juga menggerakkan beban piringan dan sensor rotary encoder. Kecepatan dan posisi relative dari piringan dapat diketahui dengan menggunakan sensor rotary encoder.

Perhatian

- Kit motor DC ini dapat bergerak dengan kecepatan cukup tinggi, sehingga jika terkena alat-alat atau anggota badan dapat menyebabkan kerusakan atau cedera.
- Pastikan tegangan yang diberikan dari power supply ke motor dibatasi pada 5 volt saja. Motor ini dapat menerima tegangan sampai 12 volt, namun putarannya akan terlalu cepat dan membahayakan.
- Jangan menyentuh komponen yang bergerak ketika motor sedang berputar.
- Bersikap serius ketika mengoperasikan motor
- Segera mematikan power supply jika terjadi masalah pada kit motor DC.
- Matikan power supply ketika melakukan penyambungan kabel-kabel.

Persiapan

- Cek fisik kit motor DC, apakah mekaniknya lengkap.
- Coba putar piringan aluminium dengan tangan, pastikan hal berikut ini:
    - poros dapat berputar
    - belt bergerak
    - sensor rotary encoder berputar
    - flexible shaft berputar
    - motor berputar
    - tidak ada mur/baut yang longgar

5.3 SENSOR ROTARY ENCODER

Tujuan

- Menguji perilaku sensor rotary encoder untuk motor DC
- Membuat sistem pengukur kecepatan
- Membuat sistem pengukur posisi

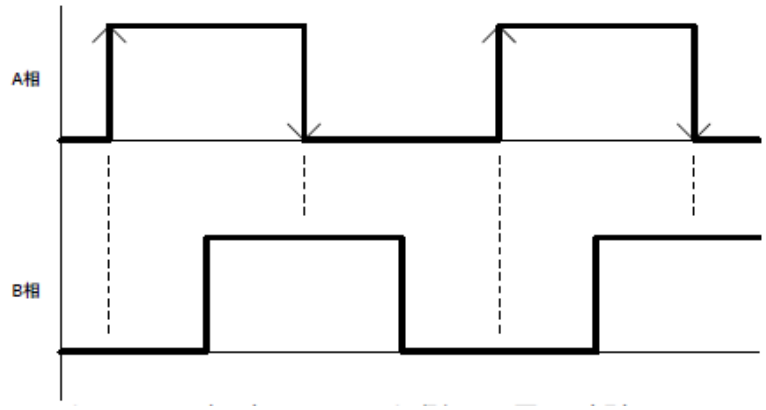
Teori

Gambar 5. 3 Rotary encoder tipe LPD3806-600BM

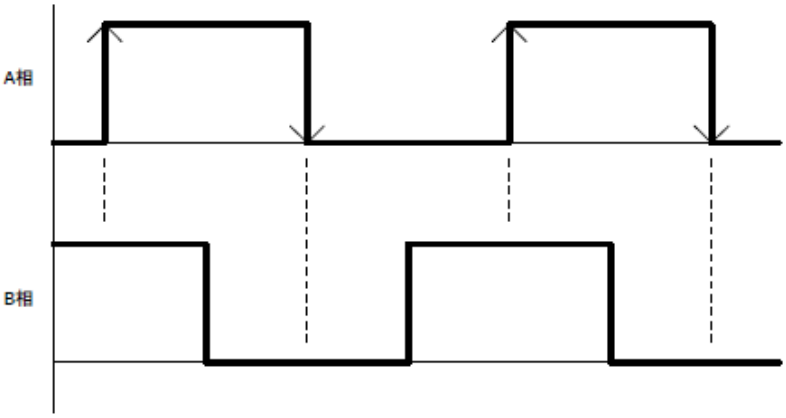
Tabel 5. 1 Kabel pada Rotary Encoder

Warna Kabel	Nama Sinyal
Merah	5 volt sampai 24 volt DC
Hitam	GND
Hijau	Sinyal A (open collector)
Putih	Sinyal B (open collector)
Shield	GND

Output rotary encoder berupa pulsa yang frekuensinya sebanding dengan kecepatan putaran. Arah putaran juga dapat diketahui dari perbedaan fasa antara sinyal A dan sinyal B. Sinyal A dan B adalah open collector, sehingga memerlukan resistor pull up. Resistor pull up ini dapat menggunakan resistor biasa, bisa juga menggunakan pull-up internal pada mikrokontroler.



Gambar 5. 4 Keluaran rotary encoder pada putaran searah jarum jam

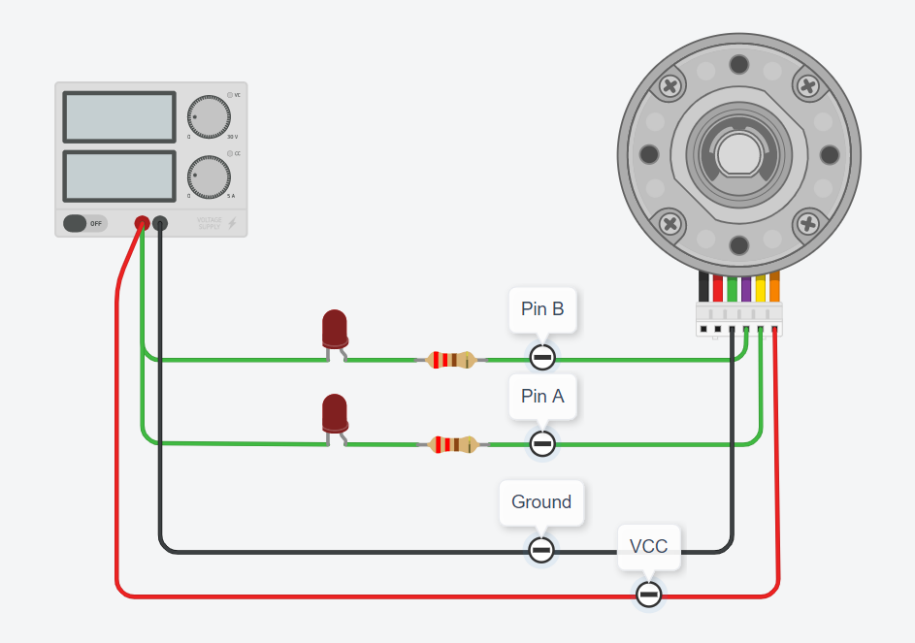


Gambar 5. 5 Keluaran rotary encoder pada putaran berlawanan jarum jam

Tugas

5.3.1 Pengujian rotary encoder

- Rangkai Rotary Encoder seperti pada rangkaian berikut:

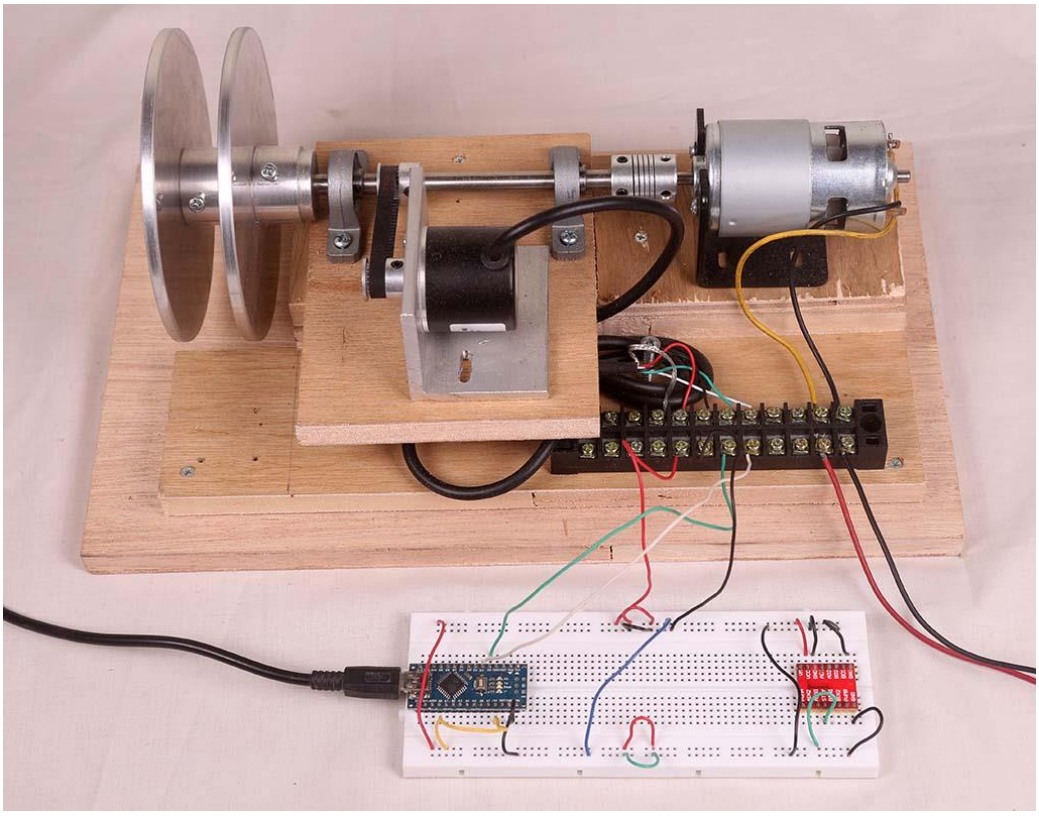


Gambar 5. 6 Rangkaian Pengujian Rotary Encoder

- Tegangan dan arus yang digunakan pada *power supply* adalah 5 V dan 1 A
- Putar secara manual poros putar Rotary Encoder dan amati perilaku lampu LED yang terpasang rangkaian
- Pastikan bahwa Rotary Encoder sudah beroperasi dengan benar sebelum melanjutkan ke percobaan-percobaan selanjutnya
- Catat hasil visualisasi pada Buku Catatan Laboratorium (BCL)
- Amati juga sinyal pada pin A dan pin B menggunakan osiloskop dan catat hasil pengamatan tersebut ke dalam BCL

5.3.2 Software Pengukur Posisi

Berikut ini foto konfigurasi untuk pengukuran posisi dan kecepatan menggunakan rotary encoder.



Gambar 5. 7 Contoh konfigurasi percobaan motor dan rotary encoder

Pada percobaan ini akan dijalankan software yang mengukur posisi sudut. Posisi awal ketika reset diasumsikan adalah 0. Output posisi ditampilkan pada port serial dengan kecepatan 9600 bps.

Tabel 5. 2 Listing program pengukur posisi

```
/*
Mengukur posisi relatif rotary encoder dengan INT0 dan INT1
*/
int state=0;
long int posisi=0;

void ISR_INT0(){
 int pinA,pinB;
 pinA=digitalRead(2);
 pinB=digitalRead(3);
 if(pinA==LOW && pinB==LOW){
  posisi--; // CCW
 }
 if(pinA==LOW && pinB==HIGH){
  posisi++;
 }
 if(pinA==HIGH&& pinB==LOW){
  posisi++; // CCW
 }
 if(pinA==HIGH && pinB==HIGH){
```

```
    posisi--; // CCW
   }
  }

  void ISR_INT1(){
   int pinA,pinB;
   pinA=digitalRead(2);
   pinB=digitalRead(3);
   if(pinA==LOW && pinB==LOW){
    posisi++; // CCW
   }
   if(pinA==LOW && pinB==HIGH){
    posisi--;
   }

   if(pinA==HIGH&& pinB==LOW){
    posisi--; // CCW
   }
   if(pinA==HIGH && pinB==HIGH){
    posisi++; // CCW
   }
  }

  void setup() {
   // initialize serial communications at 9600 bps:
   Serial.begin(9600);
   pinMode(LED_BUILTIN, OUTPUT); // untuk indikator
   pinMode(2,INPUT_PULLUP);
   pinMode(3,INPUT_PULLUP);
   attachInterrupt(digitalPinToInterrupt(2), ISR_INT0, CHANGE);
   attachInterrupt(digitalPinToInterrupt(3), ISR_INT1, CHANGE);
  }

  void loop() {
     //Konversikan hasil pengukuran rotary encoder menjadi sudut pergerakan motor
  dalam satuan radian dan kirimkan ke serial monitor
  }
```

Pertanyaan:

- Amati berapa perubahan posisi yang dicatat jika sistem diputar 360 derajat
- Software ini menggunakan pin INT0 dan INT1. Bagaimana cara menghubungkan INT0 dan INT1 supaya angka hasil pengukuran posisi sesuai (positif / negatifnya).
- Modifikasi software ini supaya dapat dijalankan menggunakan “Interupsi Pin Change”, supaya dapat dipakai di pin selain INT0 dan INT1

5.3.3 Software Pengukur Kecepatan

Buatlah software yang mengukur kecepatan sudut. Output kecepatan ditampilkan pada port serial.

Tips:

- Pengukuran kecepatan dilakukan dengan cara membagi posisi dengan waktu. Teknisnya dapat dilakukan dengan menggunakan interupsi timer untuk membangkitkan sampling dengan frekuensi tertentu, dan mereset counter posisi setiap terjadi interupsi timer. Angka posisi yang tercatat ketika terjadi interupsi mewakili kecepatan sudut.
- Interupsi Timer1 di Arduino dapat menggunakan library Timer1 (http://playground.arduino.cc/code/timer1)

Pertanyaan:

- Buatlah software yang menampilkan kecepatan sudut sistem berbasis software Arduino

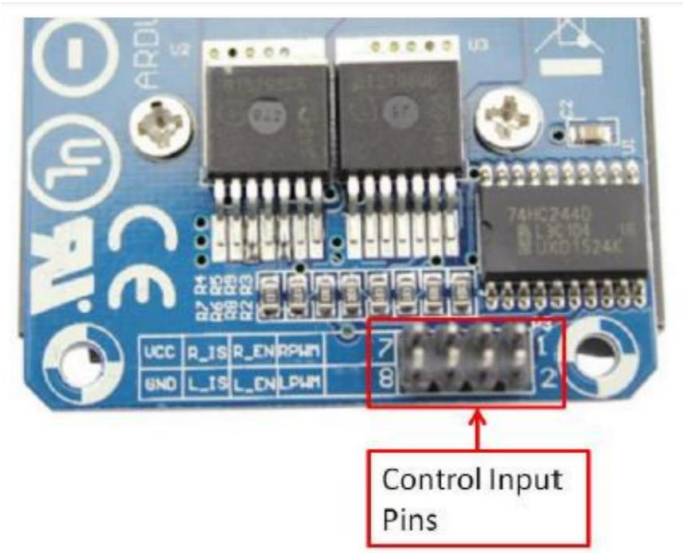
5.4 MOTOR DRIVER BTS7960 DAN KENDALI KECEPATAN OPEN LOOP

Tujuan

- Merekam output kecepatan terhadap berbagai input PWM
- Menguji perilaku Motor Driver BTS7960

Teori

Gambar 5. 8 Motor Driver BTS7960



Gambar 5. 9 Motor Driver BTS7960 (2)

Tabel 5. 3 Tabel Kebenaran motor driver

Pin No	Function	Description
1	RPWM	Forward Level or PWM signal, Active High
2	LPWM	Reverse Level or PWM signal, Active High
3	R_EN	Forward Drive Enable Input, Active High/ Low Disable
4	L_EN	Reverse Drive Enable Input, Active High/Low Disable
5	R_IS	Forward Drive, Side current alarm output
6	L_IS	Reverse Drive, Side current alarm output
7	Vcc	+5V Power Supply microcontroller
8	Gnd	Ground Power Supply microcontroller

Persiapan

- Jika konfigurasi Rotary Encoder sebelumnya (dengan menyuplai Rotary 5V dari Arduino) tidak bekerja pada percobaan ini, **hubungkan Rotary Encoder ke Adapter 12 V pada percobaan ini (artinya satu modal dengan B+)**.
- **Jangan menggunakan Power Supply Laboratoriun sebagai suplai motor DC. Gunakan adapter 12 V yang telah disediakan.**
- Hubungkan motor driver dengan konfigurasi sebagai berikut.

Pin Motor Driver	**Tujuan**
VCC	5V Arduino
GND	Ground keseluruhan
R_EN	5V Arduino
L_EN	5V Arduino
RPWM	D5 Pin Arduino

LPWM	D6 Pin Arduino
R_IS	Un-connected
L_IS	Un-Connected
B+	Positif 12 V Adapter
B-	Ground keseluruhan
M+	Positif motor
M-	Negatif motor

Tugas

- Buatlah software pada Arduino Nano untuk menjalankan motor dengan PWM -25%, -50%, -75%, 25%, 50%, 75%. (Positif negative menyatakan arah gerak motor). Manfaatkan software pengukur kecepatan yang sudah dibuat sebelumnya, catat kecepatan masing-masing PWM. Berikan analisis yang tepat dari percobaan ini!
- Lakukan pengukuran tegangan GND pada adaptor terhadap GND keluaran dari Motor Driver untuk PWM 50%.
- Amati juga keluaran Motor Driver M+ dan M- dengan menggunakan osiloskop untuk PWM 50%.

5.5 KENDALI KECEPATAN MOTOR DC DENGAN PID

Tujuan

Mengendalikan kecepatan motor DC dengan pengendali digital berbasis PID

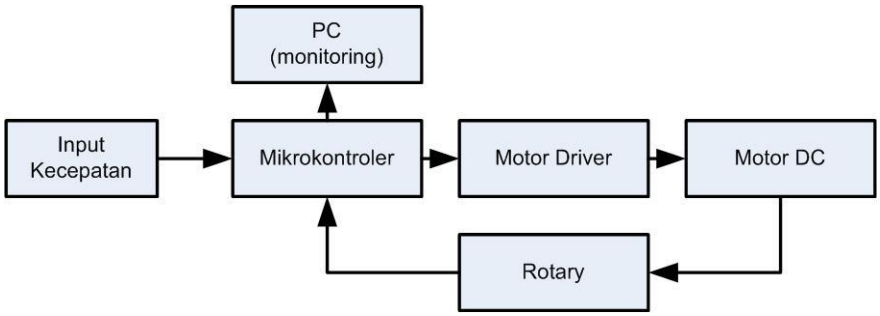
Persiapan

Buatlah rangkaian seperti pada tugas sebelumnya, lengkap dengan Rotary Encoder.

Tugas

Buatlah pengendali tipe PID (Proporsional Integral Derivatif) untuk mengendalikan kecepatan motor.

- Frekuensi sampling 1 kHz, atau disesuaikan dengan bandwidth dari sistem motor DC tersebut.
- Input kecepatan diberikan melalui set point di kode
- Data yang dikirim ke PC adalah kecepatan target dan kecepatan yang dicapai.
- Pewaktuan menggunakan interupsi timer



Gambar 5. 10 Blok diagram sistem kendali kecepatan

Hal yang dilaporkan:

- Rangkaian percobaan
- Software percobaan

PERHATIAN

Saat pertama kali dijalankan, pastikan kendali kecepatan stabil. Jika tidak stabil kecepatannya, lihat codenya lagi. Jika memang sudah benar, kemungkinan besar positif dan negative motor kurang sesuai dengan arah putaran rotary encoder. Misal: Saat diberikan PWM positif ke motor, Rotary Encoder menganggapnya negatif. Solusinya mudah, tukar saja positif negatif motor (M+ awalnya terhubung ke kabel merah motor, tukar ke kabel hitam motor, dan sebaliknya) atau tukar posisi sinyal A dan B dari rotary encoder.

5.6 KENDALI POSISI MOTOR DC DENGAN PID

Pada percobaan ini dibuat pengendali posisi motor DC dengan pengendali PID.

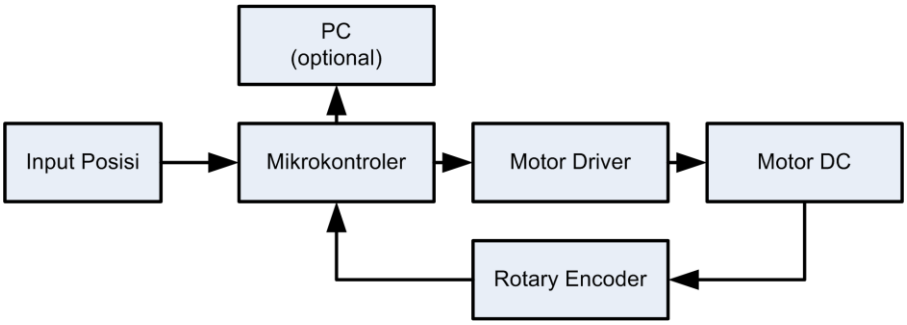
Tujuan

- Mengendalikan posisi motor DC dengan pengendali digital berbasis PID

Persiapan

Rangkaian sama dengan sistem kendali kecepatan.

Berikut diagram bloksistem kendali posisi.



Gambar 5. 11 Blok diagram sistem kendali posisi

Tugas

Buatlah pengendali tipe PID (Proporsional Integral Derivatif) untuk mengendalikan posisi motor.

- Frekuensi sampling 1 kHz, atau disesuaikan dengan bandwidth dari sistem motor DC
- Input posisi diberikan melalui port serial dari PC
- Data yang dikirim ke PC adalah posisi target dan posisi yang dicapai.
- Pewaktuan menggunakan interupsi timer

Hal yang dilaporkan:

- Rangkaian percobaan
- Software percobaan

5.7 TUGAS PENDAHULUAN

Tugas pendahuluan adalah menyiapkan semua program yang akan dijalankan.

- Modifikasi software pengukur posisi supaya dapat dijalankan menggunakan "Interupsi Pin Change", supaya dapat dipakai di pin selain INT0 dan INT1
- Modifikasi software pengukur posisi supaya dapat dijalankan pada CodeVisionAVR
- Software yang menampilkan kecepatan sudut sistem berbasis software Arduino
- Software yang menampilkan kecepatan sudut sistem berbasis software CodeVisionAVR
- Kendali Open Loop: program untuk mengirimkan sinyal PWM ke motor driver dan mencatat kecepatan motor secara periodik. (berbasis Arduino)
- Kendali Kecepatan: program kendali kecepatan berbasis PID (berbasis Arduino)
- Kendali Posisi: program kendali posisi berbasis PID (berbasis Arduino)

5.8 REFERENSI

- Arduino Nano (ATMega328) https://store.arduino.cc/usa/arduino-nano
- Rotary Encoder Teardown https://wemakethings.net/2014/05/26/rotary-encoder-teardown/
- Datasheet Rotary Encoder (bahasa Jepang) https://uamper.com/products/datasheet/LPD3806-360BM.pdf
- Reading Rotary Encoder https://playground.arduino.cc/Main/RotaryEncoders
- CodeVisionAVR http://www.hpinfotech.ro/cvavr-features.html.
- BTS7960 https://www.handsontec.com/dataspecs/module/BTS7960%20Motor%20Driver.pdf

5.9 PERTANYAAN ANALISIS

1. Jelaskan persamaan untuk mendapatkan nilai sudut terbaca oleh *rotary encoder*!
2. Jelaskan metode keamanan (pada *software*) yang digunakan dan mengapa metode tersebut dibutuhkan pada program kendali kecepatan dan kendali posisi pada percobaan ini!

5.10 DAFTAR DATA PRAKTIKUM

1. Percobaan 5.3.1 Sensor Rotary Encoder: Perilaku LED terhadap Putaran Manual pada Rotary Encoder
2. Percobaan 5.3.2 Software Pengukur Posisi: Tabel Posisi Terukur terhadap Posisi Sebenarnya
3. Percobaan 5.3.3 Software Pengukur Kecepatan: Tabel Kecepatan Terukur terhadap Kecepatan Sebenarnya
4. Percobaan 5.4 Motor Driver BTS7960 dan Kendali Kecepatan Open Loop: Tabel Kecepatan Open Loop terhadap Duty Cycle Sinyal PWM
5. Percobaan 5.4 Motor Driver BTS7960 dan Kendali Kecepatan Open Loop: Grafik Keluaran M+ dan M- pada Osiloskop
6. Percobaan 5.5 Kendali Kecepatan Motor DC dengan PID: Tabel Perbandingan Setpoint Kecepatan terhadap Kecepatan Terukur
7. Percobaan 5.6 Kendali Posisi Motor DC dengan PID: Tabel Perbandingan Setpoint Posisi terhadap Posisi Terukur

DAFTAR PUSTAKA

Datasheet ATMega 8535, www.atmel.com

APENDIKS A

PETUNJUK INSTALASI CODEVISIONAVR DAN XLOADER

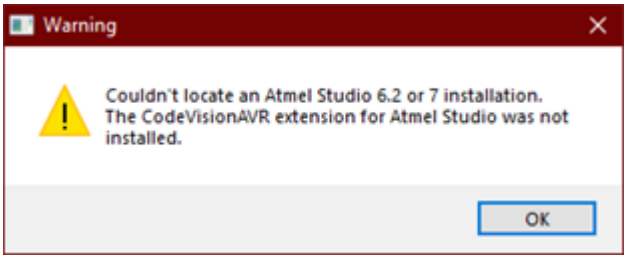
Petunjuk Install CodeVisionAVR

CodeVisionAVR adalah *integrated development environment* (IDE) dengan *source code editor* dan *compiler* untuk mikrokontroller AVR. CodeVisionAVR juga dilengkapi dengan *Automatic Program Generator*. Berikut merupakan tahapan instalasi CodeVisionAVR yang akan digunakan pada praktikum.

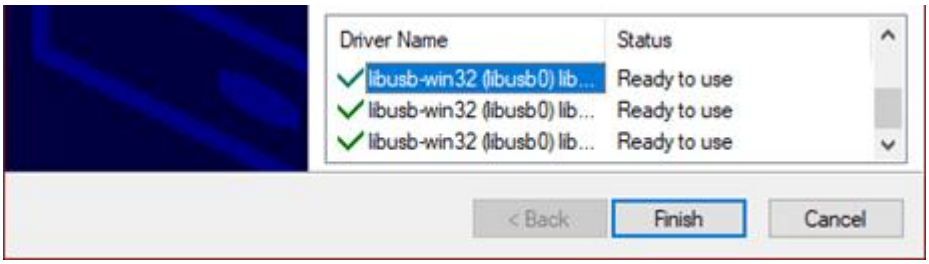
1. Buka tautan http://www.hpinfotech.ro/cvavr-download.html, kemudian unduh installer **CodeVisionAVR V3.43 Evaluation.**

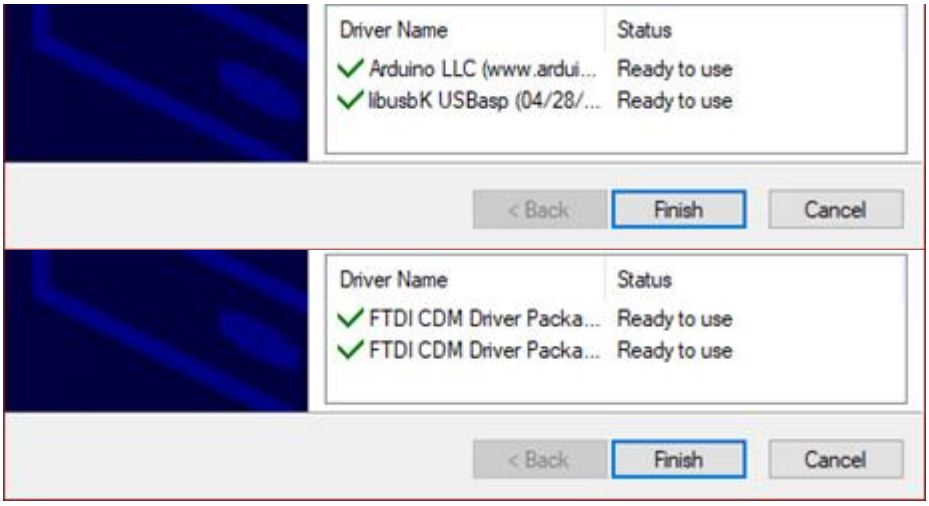
2. Ekstrak file .zip yang telah diunduh kemudian jalankan CodeVisionAVR.msi yang telah di ekstrak.

3. Ikuti tahapan pada program instalasi. Jika dalam mengikuti tahapan instalasi muncul pesan warning berikut, cukup abaikan dan klik OK.



4. Selama proses instalasi, program juga akan melakukan instalasi *driver* tambahan yang dibutuhkan. Proses instalasi tambahan akan memunculkan *Installation Wizard* tambahan. Cukup ikuti tahapan instalasi driver tambahan.

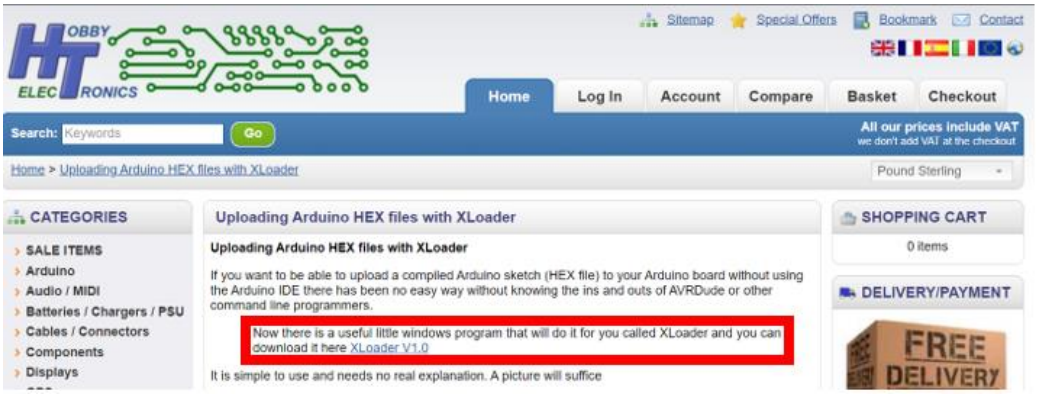




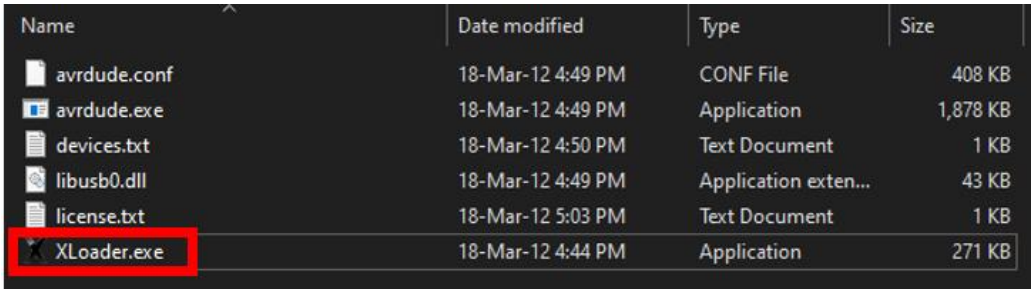
Petunjuk Install xLoader

XLoader adalah *software* yang digunakan untuk mengunggah file HEX ke Arduino board tanpa harus menggunakan IDE. Berikut merupakan tahapan instalasi xLoader yang akan digunakan pada praktikum.

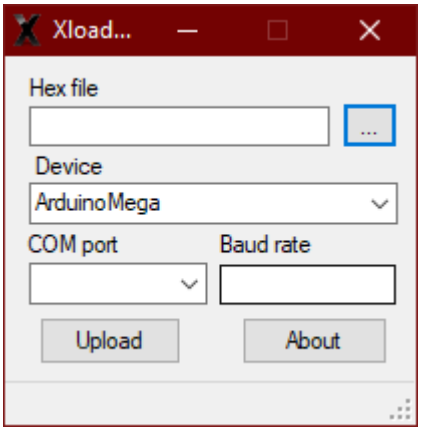
1. Buka tautan https://www.hobbytronics.co.uk/arduino-xloader, kemudian unduh XLoader V1.0.



2. Ekstrak file .zip yang telah diunduh kemudian jalankan Xloader.exe dari folder yang telah di ekstrak.



3. XLoader siap digunakan.


Xload...
Hex file
...
Device
ArduinoMega
COM port
Baud rate
Upload
About

APENDIKS B

TUTORIAL ESP-IDF

Versi lengkap dengan Bahasa Inggris dapat dilihat pada link berikut ini. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/
Contoh (*example*) dan *source code*dari ESP-IDF dapat dilihat pada link berikut ini. https://github.com/espressif/esp-idf/tree/release/v4.2

Tutorial ini berisi cara melakukan instalasi ESP-IDF, menggunakan *example* yang disediakan ESP-IDF, dan membuat proyek ESP-IDF sederhana dari awal.

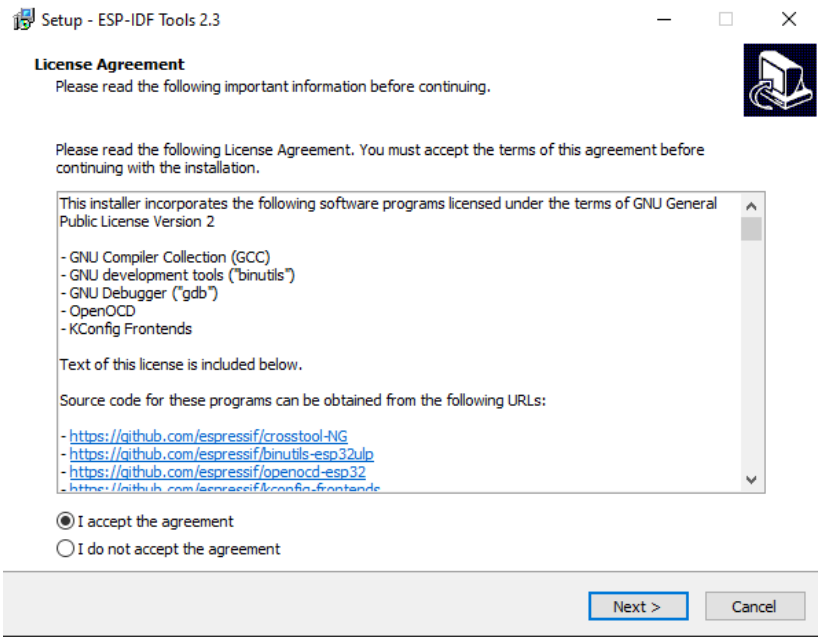
Pastikan bahwa terdapat koneksi internet sebelum melakukan semua tahap-tahap instalasi ESP-IDF.

Proses Instalasi

A. Instalasi pada Windows:

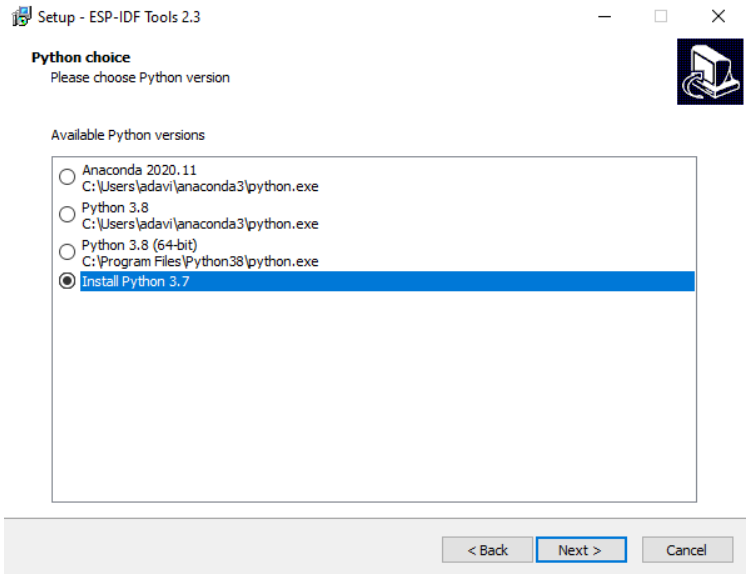
1. Unduh (*download*) ESP-IDF Tools Installer pada link berikut. (343 MB) https://dl.espressif.com/dl/esp-idf-tools-setup-2.3.exe

2. Lakukan instalasi dengan membuka file esp-idf-tools-setup-2.3.exe dan terbuka halaman seperti pada gambar di bawah ini. Pilih *I accept the agreement* kemudian lanjut dengan menekan tombol *Next*.



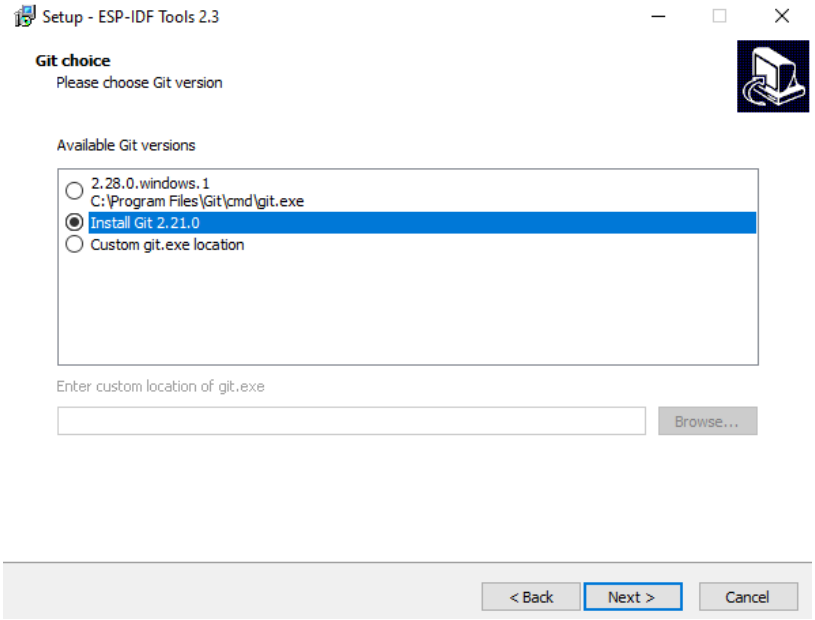
Gambar x Halaman Pertama Setup ESP-IDF Tools 2.3

3. Setelah menekan tombol *Next*, dapat dilakukan pemilihan untuk versi python yang akan digunakan. Jika belum pernah dilakukan instalasi python pada perangkat yang digunakan, dapat memilih pilihan *Install Python 3.7*. Jika ingin menggunakan versi python yang sudah pernah diinstal, sangat direkomendasikan untuk menggunakan Python versi 3.6 hingga 3.8. (sangat disarankan untuk tidak menggunakan Python versi 3.6 kebawah dan Python 3.9).



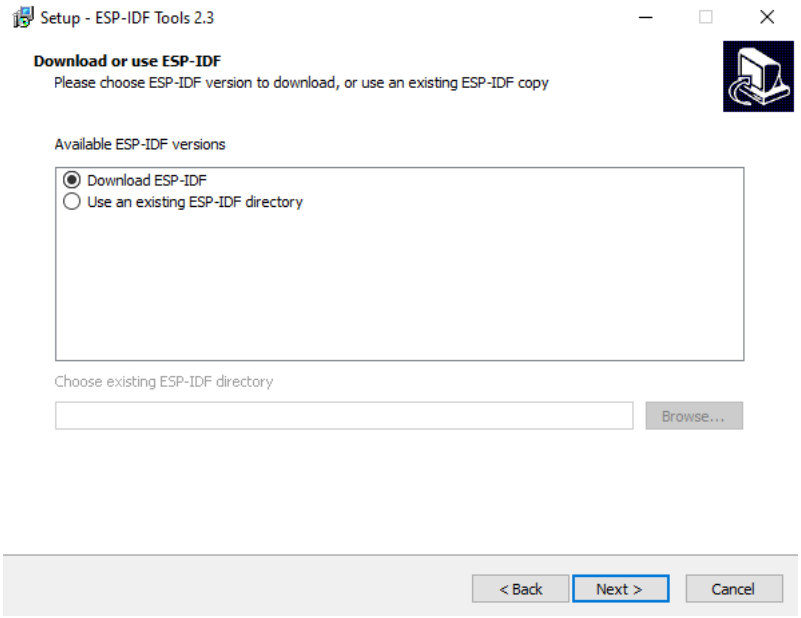
Gambar x Halaman Kedua Setup ESP-IDF Tools 2.3

4. Setelah menekan tombol *Next* pada halaman kedua, akan muncul halaman ketiga yaitu halaman pemilihan Git seperti pada gambar di bawah ini. Jika belum pernah melakukan instalasi Git pada perangkat yang digunakan, pilih *Install Git 2.21.0*. Jika ingin menggunakan Git yang sudah pernah diinstal, silahkan pilih Git yang diinginkan.



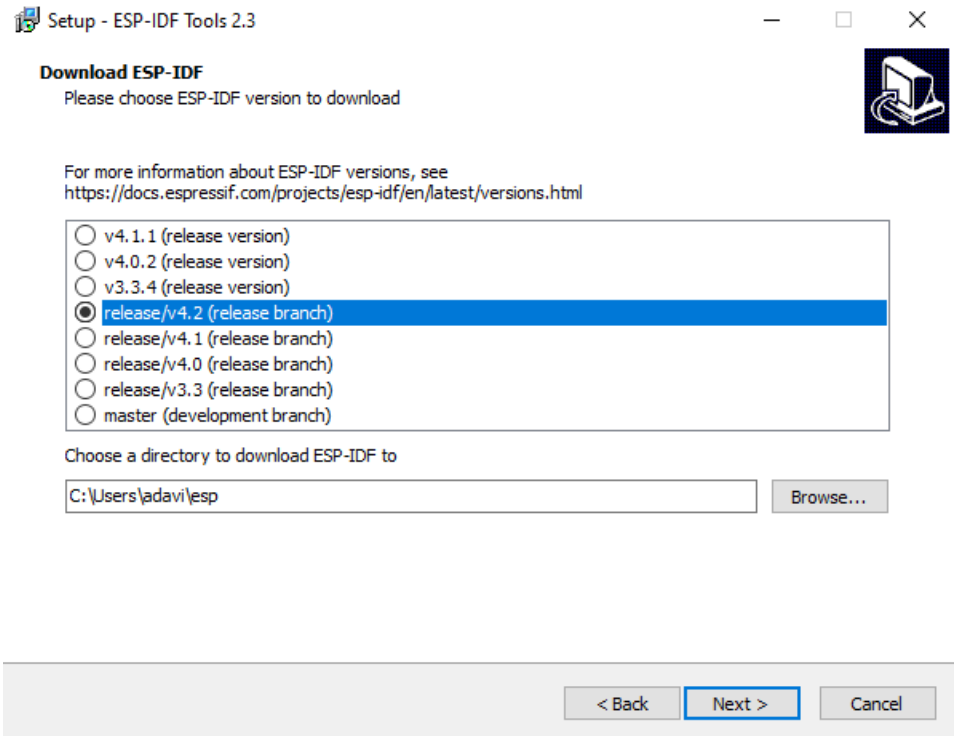
Gambar x Halaman Ketiga Setup ESP-IDF Tools 2.3

5. Setelah menekan tombol *Next* pada halaman ketiga, akan muncul halaman keempat seperti pada gambar di bawah ini. Pilih *Download ESP-IDF* kemudian tekan tombol *Next*.



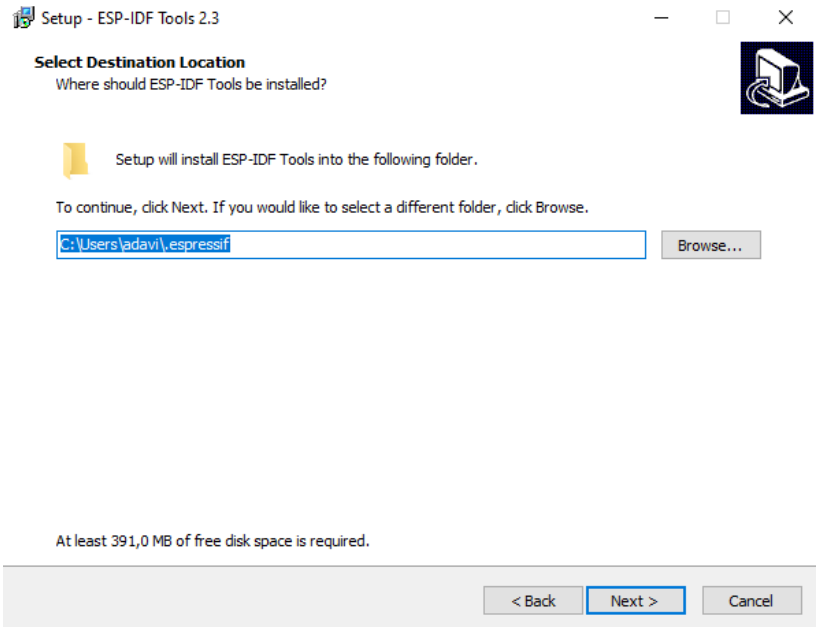
Gambar x Halaman Keempat Setup ESP-IDF Tools 2.3

6. Setelah menekan tombol *Next* pada halaman keempat, pilih versi stable ESP-IDF terbaru (v4.2, sewaktu penulisan Februari 2021) pada halaman kelima seperti pada gambar di bawah ini. Setelah itu, pilih folder tempat instalasi ESP-IDF yaitu C:\Users\%USERNAME%\esp (pada contoh gambar, %username% perangkat yang digunakan adalah adavi). Jika belum membuat folder tersebut, dapat dilakukan pada Command Prompt dengan perintah `mkdir %userprofile%\esp` atau dengan membuat folder tersebut secara manual.
   *Catatan: %USERNAME% tidak boleh memiliki spasi



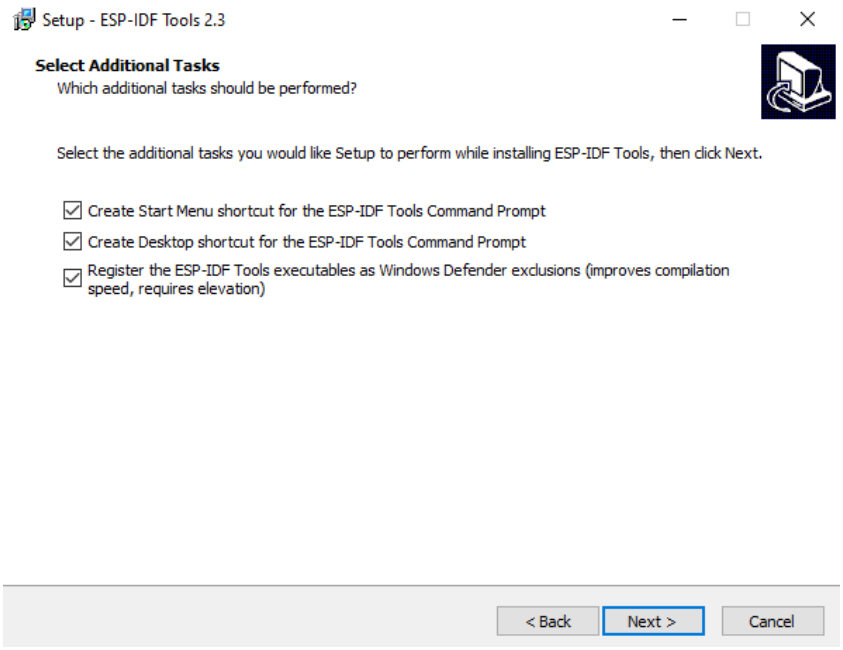
Gambar x Halaman Kelima Setup ESP-IDF Tools 2.3

7. Setelah menekan tombol *Next* pada halaman kelima, akan muncul halaman keenam seperti pada gambar di bawah ini. Langsung saja tekan tombol *Next*.



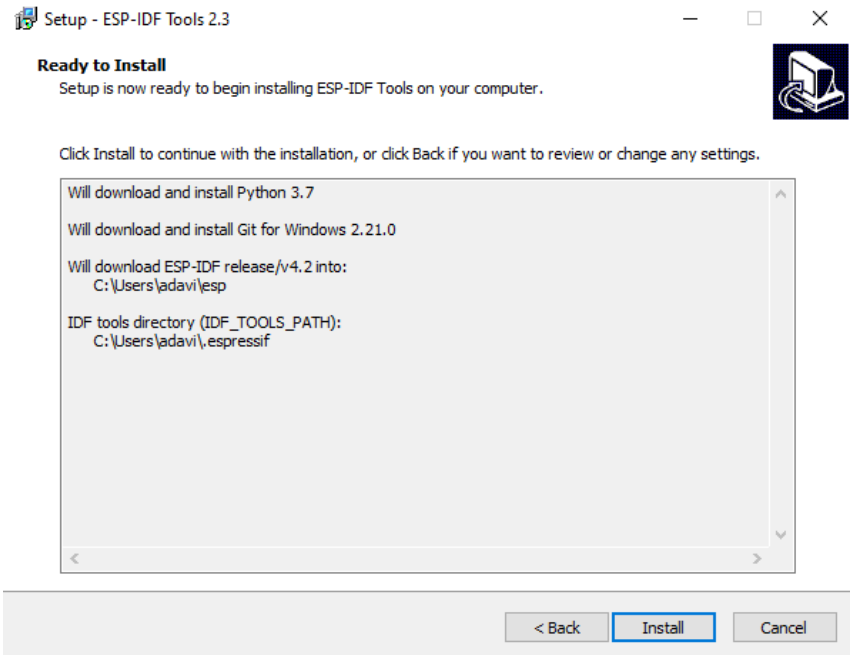
Gambar x Halaman Keenam Setup ESP-IDF Tools 2.3

8. Pada halaman ketujuh ini disarankan untuk mencentang poin ke-3 agar mempercepat waktu kompilasi. Pembuatan *Start menu* dan *desktop shortcut* dibebaskan kepada praktikan (jika mengalami kebingungan, centang semua saja). Setelah itu tekan tombol *Next*.

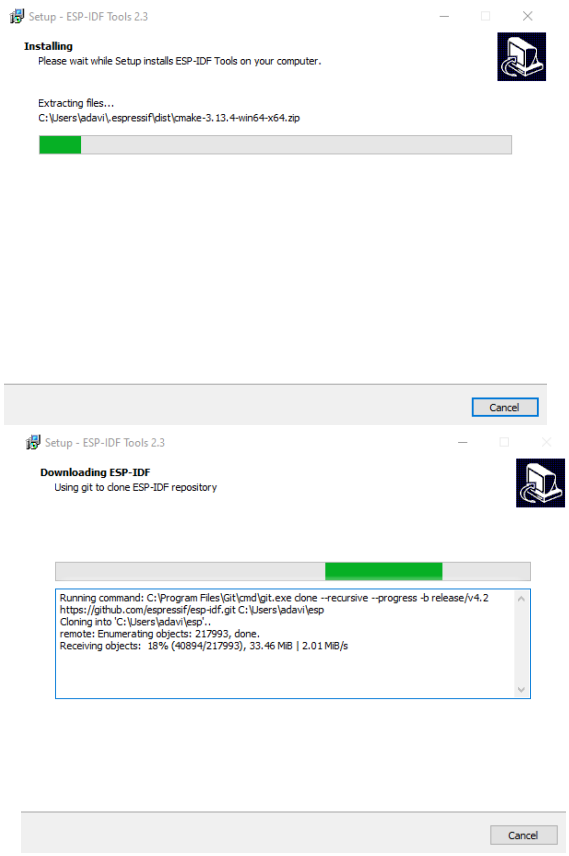


Gambar x Halaman ketujuh Setup ESP-IDF Tools 2.3

9. Setelah menekan tombol *Next* pada halaman ketujuh, proses instalasi akan memasuki halaman terakhir yang berisi rangkuman dari instalasi yang akan dilakukan. Jika belum merasa semua sudah benar, lakukan pengecekan ulang menggunakan tombol *Back*. Jika merasa semua sudah benar, maka tekan tombol *Install* dan proses instalasi akan berjalan.

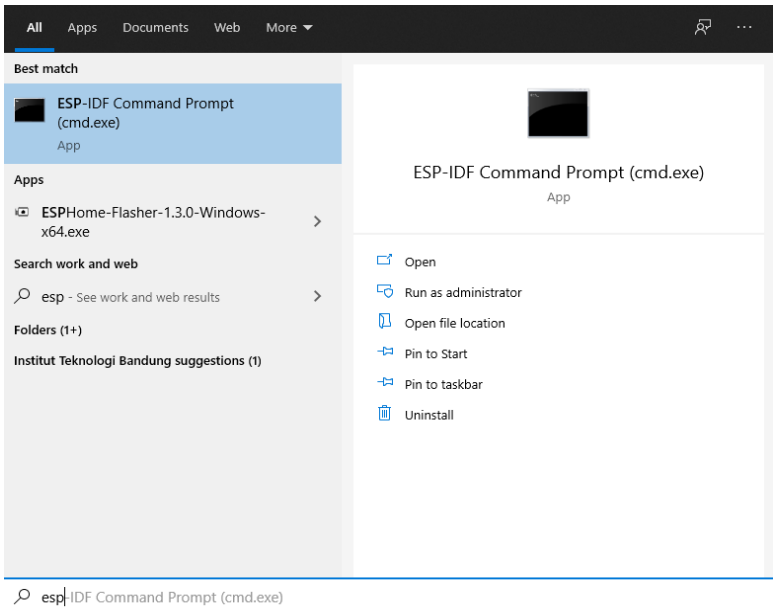


Gambar x Halaman kedelapan Setup ESP-IDF Tools 2.3



10. Setelah proses instalasi selesai, terdapat 2 cara dalam menggunakan ESP-IDF, yaitu:

    a. **(Recommended)** Cara pertama yaitu membuka ESP-IDF Command Prompt melalui Start. Jika instalasi berhasil, dapat dilihat hasil seperti pada gambar-gambar di bawah ini.



Gambar x ESP-IDF Command Prompt pada Start

Gambar x Environment Variables yang sudah di-setup oleh ESP-IDF Command Prompt.

b. **(Advanced)** Cara kedua yaitu membuka Command Prompt kemudian mengetikkan `%userprofile%\esp\esp-idf\export.bat` pada Command Prompt tersebut. (Dengan kondisi folder esp-idf terletak pada folder %userprofile%/esp).

B. Instalasi pada macOS / OS X:
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/#
Pilih versi stable terbaru yaitu ESP-IDF v4.2 (sewaktu penulisan Februari 2021)

C. Instalasi pada Linux:
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/#
Pilih versi stable terbaru yaitu ESP-IDF v4.2 (sewaktu penulisan Februari 2021)

Cara menggunakan example yang tersedia pada ESP-IDF

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html#step-5-start-a-project

https://github.com/espressif/esp-idf/tree/526f682/examples

*Catatan: Contoh gambar command prompt menggunakan folder desktop dengan sebagai pengganti folder esp.

*Catatan: Pertama kali melakukan build membutuhkan waktu yang cukup lama (build dari awal dan juga setelah melakukan *fullclean*). Namun jika melakukan build pada program yang sudah pernah melakukan build hanya membutuhkan waktu sebentar.

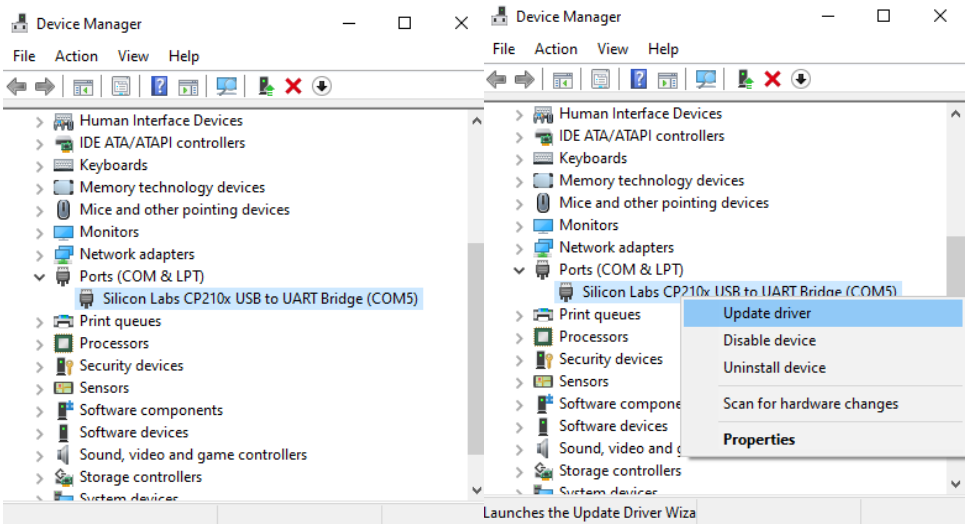
1. Buka Command Prompt / Terminal dengan *environment* ESP-IDF.
   Windows: Buka ESP-IDF Command Prompt.
   Linux & macOS: pada terminal yang akan digunakan ketik
   `. $HOME/esp/esp-idf/export.sh` atau `. $HOME/esp/esp-idf/export.fish`

```
ESP-IDF Command Prompt (cmd.exe) - "C:\Users\adavi\.espressif\idf_cmd_init.bat" "C:\Program Files\Python38\" "C:\Program Files\Git\cmd\"
Using Python in C:\Program Files\Python38\
Python 3.8.7
Using Git in C:\Program Files\Git\cmd\
git version 2.28.0.windows.1
Setting IDF_PATH: C:\Users\adavi\Desktop\esp-idf

Adding ESP-IDF tools to PATH...
    C:\Users\adavi\.espressif\tools\xtensa-esp32-elf\esp-2019r2-8.2.0\xtensa-esp32-elf\bin
    C:\Users\adavi\.espressif\tools\esp32ulp-elf\2.28.51.20170517\esp32ulp-elf-binutils\bin
    C:\Users\adavi\.espressif\tools\cmake\3.13.4\bin
    C:\Users\adavi\.espressif\tools\openocd-esp32\v0.10.0-esp32-20190313\openocd-esp32\bin
    C:\Users\adavi\.espressif\tools\mconf\v4.6.0.0-idf-20190628\
    C:\Users\adavi\.espressif\tools\ninja\1.9.0\
    C:\Users\adavi\.espressif\tools\idf-exe\1.0.1\
    C:\Users\adavi\.espressif\tools\ccache\3.7\
    C:\Users\adavi\.espressif\python_env\idf4.0_py3.8_env\Scripts
    C:\Users\adavi\Desktop\esp-idf\tools

Checking if Python packages are up to date...
Python requirements from C:\Users\adavi\Desktop\esp-idf\requirements.txt are satisfied.

Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

  idf.py build


C:\Users\adavi\Desktop\esp-idf>
```

2. Pindahkan ke folder *example* yang diinginkan (misalnya *example* Hello World) dengan mengetikkan pada Command Prompt / Terminal.
   Windows: `cd %userprofile%/esp/esp-idf/examples/get-started/hello_world`
   Linux & macOS: `cd ~/esp/esp-idf/examples/get-started/hello_world`

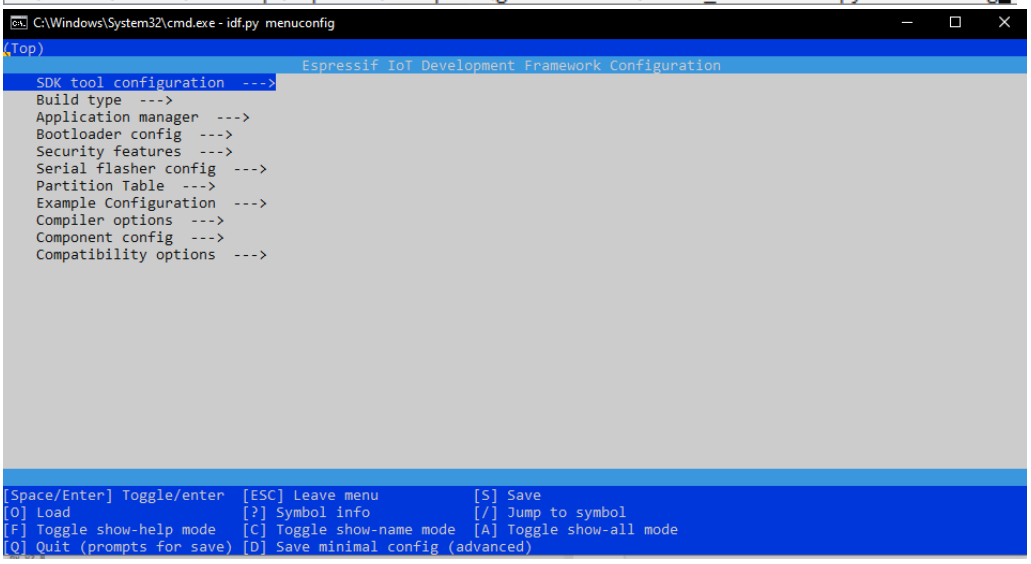
```
C:\Users\adavi\Desktop\esp-idf>cd %userprofile%\desktop\esp-idf\examples\get-started\hello_world

C:\Users\adavi\Desktop\esp-idf\examples\get-started\hello_world>
```

3. Hubungkan ESP32 dengan Laptop/PC via kabel Micro USB. Buka Device Manager melalui Start dan pastikan board ESP32 sudah terbaca, jika belum terbaca, lakukan update driver. (Driver akan otomatis di-download pada saat pertama kali menghubungkan ESP32 dengan Laptop/PC)



4. Jika ESP32 sudah terbaca seperti pada nomor 3, akan dilakukan konfigurasi dengan memasukkan perintah `idf.py menuconfig` pada folder Hello World. Setelah itu, tunggu beberapa saat hingga muncul menu seperti gambar berikut. Navigasi dilakukan dengan *arrow key* dan *enter*. Namun, untuk Hello World tidak membutuhkan konfigurasi apapun sehingga dapat memilih menu exit dengan panah kanan dan kemudian tekan *enter*. ***Hanya untuk pengetahuan tambahan, tidak digunakan pada praktikum**

```
C:\Users\adavi\Desktop\esp-idf\examples\get-started\hello_world>idf.py menuconfig
```



5. Setelah melakukan konfigurasi, dapat dilakukan build project dengan cara mengetikkan perintah `idf.py build`. Setelah itu, akan muncul progress build yang sedang berlangsung. Pada saat sedang melakukan *build*, CPU laptop/PC akan digunakan dengan cukup keras sehingga disarankan untuk tidak menjalankan program lain yang juga membutuhkan CPU dalam porsi besar (Karena *built* dapat dikatakan sebagai proses kompilasi).

```
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/adavi/Desktop/esp-idf/examples/get-started/hello_world/build
[203/827] Building C object esp-idf/bootloader_support/CMakeFiles/__idf_bootloader_support.dir/src/bootloader_clock.c.obj
```

6. Setelah selesai *build*, akan muncul notifikasi seperti pada gambar berikut ini yang menandakan bahwa file .bin sukses dibuat.

```
[60/62] Linking C static library esp-idf\main\libmain.a
[61/62] Linking C executable bootloader.elf
[62/62] Generating binary image from built executable
esptool.py v2.8
Generated C:/Users/adavi/Desktop/esp-idf/examples/get-started/hello_world/build/bootloader/bootloader.bin
[827/827] Generating binary image from built executable
esptool.py v2.8
Generated C:/Users/adavi/Desktop/esp-idf/examples/get-started/hello_world/build/hello-world.bin

Project build complete. To flash, run this command:
C:\Users\adavi\.espressif\python_env\idf4.0_py3.8_env\Scripts\python.exe ..\..\..\components\esptool_py\esptool\espto
ol.py -p (PORT) -b 460800 --before default_reset --after hard_reset write_flash --flash_mode dio --flash_size detect
--flash_freq 40m 0x1000 build\bootloader\bootloader.bin 0x8000 build\partition_table\partition-table.bin 0x10000 buil
d\hello-world.bin
or run 'idf.py -p (PORT) flash'

C:\Users\adavi\Desktop\esp-idf\examples\get-started\hello_world>
```

7. Kemudian, akan dilakukan *flashing* file firmware binary (.bin) tersebut ke ESP32 dengan cara memasukkan perintah `idf.py -p PORT flash`. PORT merupakan nama USB Port dimana ESP32 tersambung. Dapat dilihat pada nomor 3 bahwa ESP32 memiliki nama port COM5, sehingga PORT diganti dengan COM5 seperti pada gambar berikut.

```
C:\Users\adavi\Desktop\esp-idf\examples\get-started\hello_world>idf.py -p COM5 flash
```

8. Saat sedang melakukan *flashing* akan muncul `Connecting...` seperti pada gambar berikut. Jika hal tersebut terjadi, maka tekan tombol *BOOT* pada board ESP32 selama beberapa saat hingga `Connecting...` berhenti dan terdapat keluaran tambahan seperti pada gambar kedua berikut. (ESP32 memiliki 2 push button yaitu *BOOT* dan *EN*)

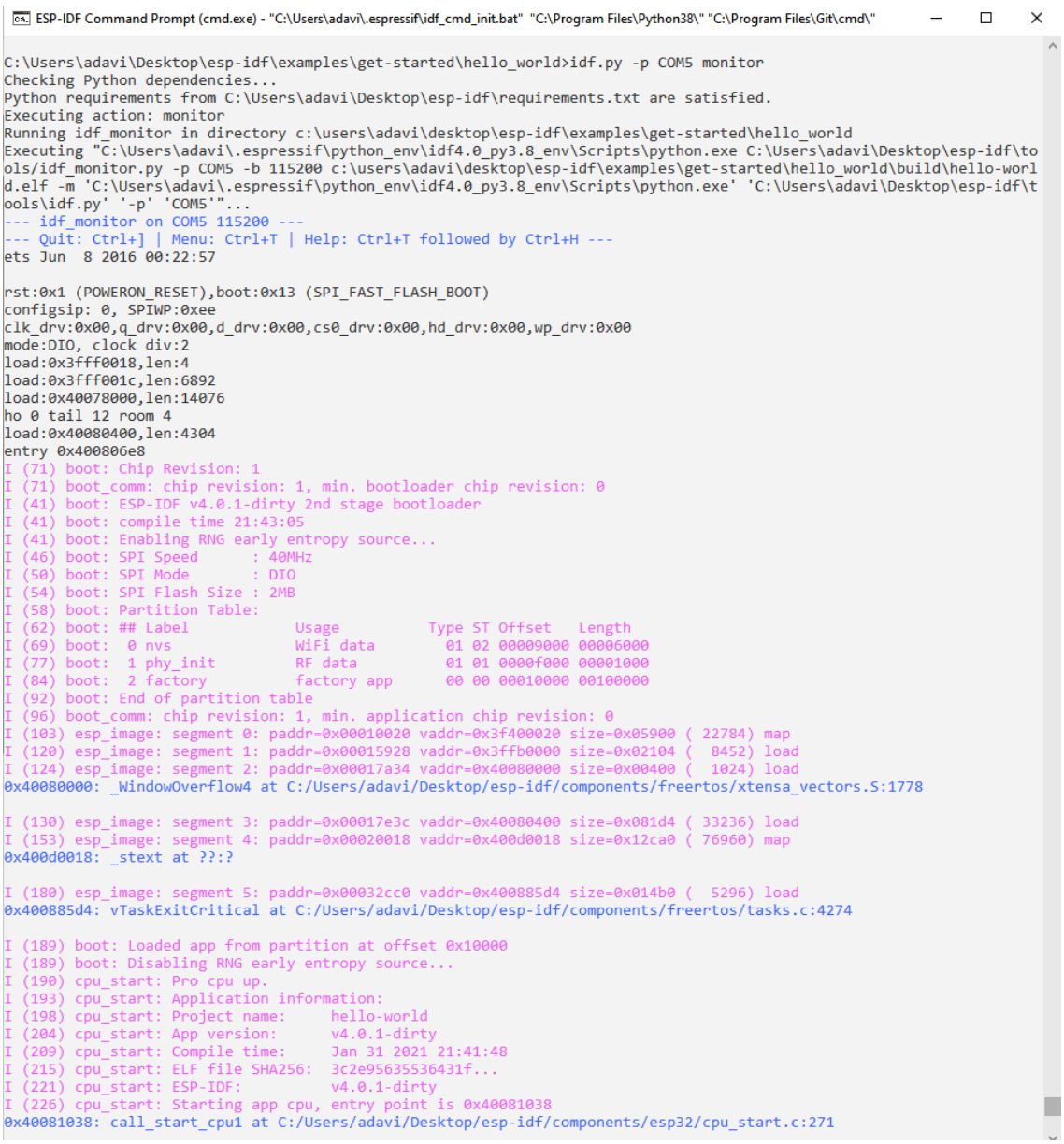
```
C:\Users\adavi\Desktop\esp-idf\examples\get-started\hello_world>idf.py -p COM5 flash
Checking Python dependencies...
Python requirements from C:\Users\adavi\Desktop\esp-idf\requirements.txt are satisfied.
Adding flash's dependency "all" to list of actions
Executing action: all (aliases: build)
Running ninja in directory c:\users\adavi\desktop\esp-idf\examples\get-started\hello_world\build
Executing "ninja all"...
[1/4] cmd.exe /C "cd /D C:\Users\adavi\Desktop\esp-idf\ex...**********************************************************"
Partition table binary generated. Contents:
*******************************************************************************
# Espressif ESP32 Partition Table
# Name, Type, SubType, Offset, Size, Flags
nvs,data,nvs,0x9000,24K,
phy_init,data,phy,0xf000,4K,
factory,app,factory,0x10000,1M,
*******************************************************************************
[2/4] Performing build step for 'bootloader'
ninja: no work to do.
Executing action: flash
Running esptool.py in directory c:\users\adavi\desktop\esp-idf\examples\get-started\hello_world\build
Executing "C:\Users\adavi\.espressif\python_env\idf4.0_py3.8_env\Scripts\python.exe C:\Users\adavi\Desktop\esp-idf\co
mponents/esptool_py/esptool/esptool.py -p COM5 -b 460800 --before default_reset --after hard_reset write_flash @flash
_project_args"...
esptool.py -p COM5 -b 460800 --before default_reset --after hard_reset write_flash --flash_mode dio --flash_freq 40m
--flash_size 2MB 0x8000 partition_table/partition-table.bin 0x1000 bootloader/bootloader.bin 0x10000 hello-world.bin
esptool.py v2.8
Serial port COM5
Connecting........_____.
```

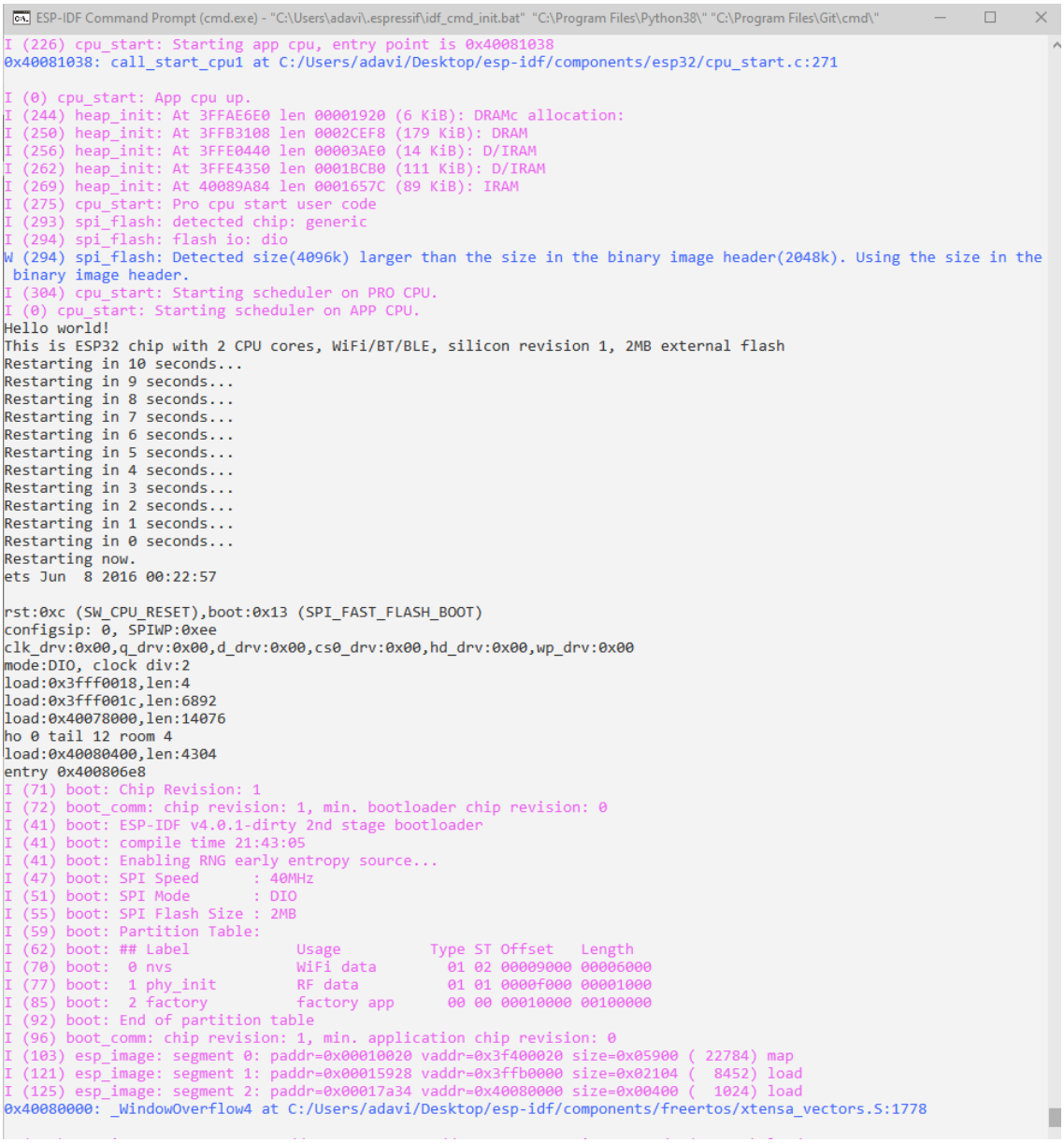
```
Connecting........_____.
Detecting chip type... ESP32
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 24:0a:c4:59:1c:a0
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Compressed 3072 bytes to 103...
Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.0 seconds (effective 529.7 kbit/s)...
Hash of data verified.
Compressed 25376 bytes to 14959...
Wrote 25376 bytes (14959 compressed) at 0x00001000 in 0.4 seconds (effective 539.8 kbit/s)...
Hash of data verified.
Compressed 147872 bytes to 76813...
Wrote 147872 bytes (76813 compressed) at 0x00010000 in 1.9 seconds (effective 617.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
Done

C:\Users\adavi\Desktop\esp-idf\examples\get-started\hello_world>
```

9. Jika output yang didapat sama seperti gambar diatas, maka program berhasil di-*flash* ke ESP32. Selanjutnya kita ingin mengakses serial communication dengan ESP32 menggunakan IDF Monitor dengan cara mengetikkan perintah `idf.py -p PORT monitor`. Jangan lupa untuk mengganti PORT dengan COM PORT ESP32 seperti pada gambar berikut ini.

```
C:\Users\adavi\Desktop\esp-idf\examples\get-started\hello_world>idf.py -p COM5 monitor
```

Akan muncul keluaran seperti berikut. Dapat dilihat bahwa program akan memunculkan informasi dari board ESP32 yang sedang digunakan dan menampilkan tulisan Hello World. Kemudian tunggu 10 detik dan melakukan restart ESP32 melalui program.


ESP-IDF Command Prompt (cmd.exe) - "C:\Users\adavi\.espressif\idf_cmd_init.bat" "C:\Program Files\Python38\" "C:\Program Files\Git\cmd\"
C:\Users\adavi\Desktop\esp-idf\examples\get-started\hello_world>idf.py -p COM5 monitor
Checking Python dependencies...
Python requirements from C:\Users\adavi\Desktop\esp-idf\requirements.txt are satisfied.
Executing action: monitor
Running idf_monitor in directory c:\users\adavi\desktop\esp-idf\examples\get-started\hello_world
Executing "C:\Users\adavi\.espressif\python_env\idf4.0_py3.8_env\Scripts\python.exe C:\Users\adavi\Desktop\esp-idf\tools/idf_monitor.py -p COM5 -b 115200 c:\users\adavi\desktop\esp-idf\examples\get-started\hello_world\build\hello-world.elf -m 'C:\Users\adavi\.espressif\python_env\idf4.0_py3.8_env\Scripts\python.exe' 'C:\Users\adavi\Desktop\esp-idf\tools\idf.py' '-p' 'COM5'"...
--- idf_monitor on COM5 115200 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
ets Jun 8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:6892
load:0x40078000,len:14076
ho 0 tail 12 room 4
load:0x40080400,len:4304
entry 0x400806e8
I (71) boot: Chip Revision: 1
I (71) boot_comm: chip revision: 1, min. bootloader chip revision: 0
I (41) boot: ESP-IDF v4.0.1-dirty 2nd stage bootloader
I (41) boot: compile time 21:43:05
I (41) boot: Enabling RNG early entropy source...
I (46) boot: SPI Speed : 40MHz
I (50) boot: SPI Mode : DIO
I (54) boot: SPI Flash Size : 2MB
I (58) boot: Partition Table:
I (62) boot: ## Label Usage Type ST Offset Length
I (69) boot: 0 nvs WiFi data 01 02 00009000 00006000
I (77) boot: 1 phy_init RF data 01 01 0000f000 00001000
I (84) boot: 2 factory factory app 00 00 00010000 00100000
I (92) boot: End of partition table
I (96) boot_comm: chip revision: 1, min. application chip revision: 0
I (103) esp_image: segment 0: paddr=0x00010020 vaddr=0x3f400020 size=0x05900 (22784) map
I (120) esp_image: segment 1: paddr=0x00015928 vaddr=0x3ffb0000 size=0x02104 (8452) load
I (124) esp_image: segment 2: paddr=0x00017a34 vaddr=0x40080000 size=0x00400 (1024) load
0x40080000: _WindowOverflow4 at C:/Users/adavi/Desktop/esp-idf/components/freertos/xtensa_vectors.S:1778
I (130) esp_image: segment 3: paddr=0x00017e3c vaddr=0x40080400 size=0x081d4 (33236) load
I (153) esp_image: segment 4: paddr=0x00020018 vaddr=0x400d0018 size=0x12ca0 (76960) map
0x400d0018: _stext at ??:?
I (180) esp_image: segment 5: paddr=0x00032cc0 vaddr=0x400885d4 size=0x014b0 (5296) load
0x400885d4: vTaskExitCritical at C:/Users/adavi/Desktop/esp-idf/components/freertos/tasks.c:4274
I (189) boot: Loaded app from partition at offset 0x10000
I (189) boot: Disabling RNG early entropy source...
I (190) cpu_start: Pro cpu up.
I (193) cpu_start: Application information:
I (198) cpu_start: Project name: hello-world
I (204) cpu_start: App version: v4.0.1-dirty
I (209) cpu_start: Compile time: Jan 31 2021 21:41:48
I (215) cpu_start: ELF file SHA256: 3c2e95635536431f...
I (221) cpu_start: ESP-IDF: v4.0.1-dirty
I (226) cpu_start: Starting app cpu, entry point is 0x40081038
0x40081038: call_start_cpu1 at C:/Users/adavi/Desktop/esp-idf/components/esp32/cpu_start.c:271



10. Selamat!! Anda sukses melakukan *build*, *config*, *flash*, dan *monitor* menggunakan ESP32.

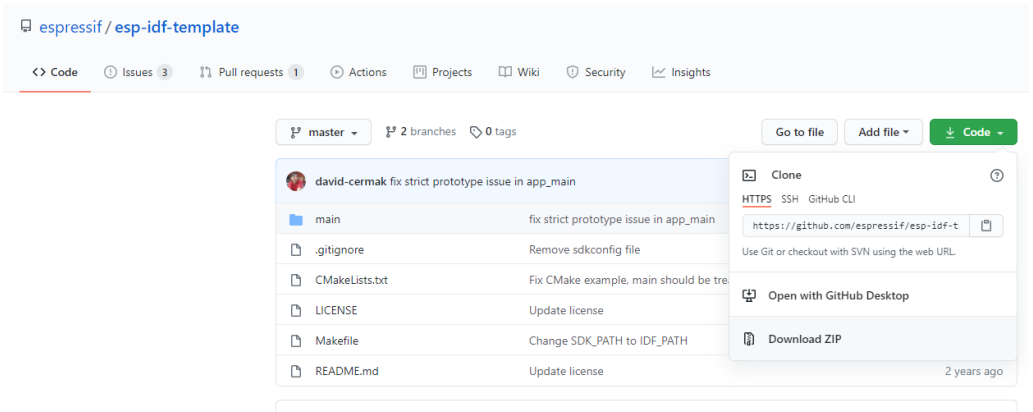
Jika anda mencapai tahap ini selamat, anda sudah mengenal dan dapat menggunakan fitur-fitur dasar dari ESP-IDF.

Jika ingin membuat project baru, silahkan *copy and paste* folder *example* yang sudah ada ke tempat yang anda inginkan. Setelah itu anda dapat mengubah file-file yang diinginkan. Namun ada baiknya jika anda dapat membuat project sederhana dari template project yang sudah tersedia seperti pada tutorial selanjutnya.

(Advanced) Membuat Program dengan ESP-IDF tanpa menggunakan example

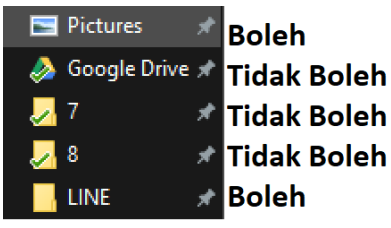
*Jika anda dapat menyelesaikan ini, akan sangat membantu pengerjaan praktikum yang menggunakan ESP-IDF.

1. Download zip pada link berikut. https://github.com/espressif/esp-idf-template



2. Extract pada folder yang diinginkan (tidak harus berada di dalam folder ESP-IDF). Folder tidak boleh berada di dalam kondisi *backup and sync* seperti google drive atau onedrive seperti pada gambar di bawah ini. *Folder path* juga tidak boleh memiliki spasi. Contoh:

"D:\Documents\PraktikumSisMik" ✔;
"D:\Documents\Prak-Sis-Mik" ✔;
"D:\Documents\Prak_Sis_Mik" ✔;
"D:\Documents\Prak Sis Mik" ✗;
"D:\Documents\Semester 6\PrakSisMik"✗.



3. Ubah nama folder menjadi nama yang diinginkan.

4. Melalui ESP-IDF Command Prompt, masuk ke folder tersebut

5. Buka file main.c pada folder main dan ubah kode di dalam main.c menjadi https://github.com/espressif/esp-idf/blob/master/examples/get-started/blink/main/blink_example_main.c

6. Jika ingin mengubah pin LED langsung pada file main.c (**akan** diterapkan pada praktikum), ubah `CONFIG_BLINK_GPIO` pada line 18 menjadi angka `2` dan langkahi nomor 7 dan 8. Jika ingin mengubah pin LED melalui `idf.py menuconfig` (**tidak akan** diterapkan pada praktikum), lakukan nomor 7 dan 8.

7. Buka file Kconfig.projbuild menggunakan *text editor* dan ubah menjadi https://github.com/espressif/esp-idf/blob/master/examples/get-started/blink/main/Kconfig.projbuild

8. Masuk ke menuconfig (`idf.py menuconfig`) dan pilih *Example Configuration* dan ubah pin GPIO tersebut dari pin 5 menjadi pin 2 (*Built-In LED* pada DOIT ESP32 Devkit V1).

9. Build dan flash dapat dilakukan sekaligus dengan perintah `idf.py flash`. Namun harus dipastikan bahwa ESP32 dalam keadaan terhubung dan terdeteksi oleh laptop/PC. (Informasi tambahan: Jika tidak menggunakan `-p PORT`, maka ESP32 akan mencari COM PORT secara otomatis)

APENDIKS C

TABEL ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	169	_	185	╣	201	╔	217	┘	233	⊕	249	.
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	.
138	è	154	Ü	171	½	187	╗	203	╦	219	▉	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	_
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	▒	192	└	208	╨	224	α	240	≡		

Source: www.LookupTables.com


INSTITUT TEKNOLOGI BANDUNG
1920


INSTITUT TEKNOLOGI BANDUNG
1920