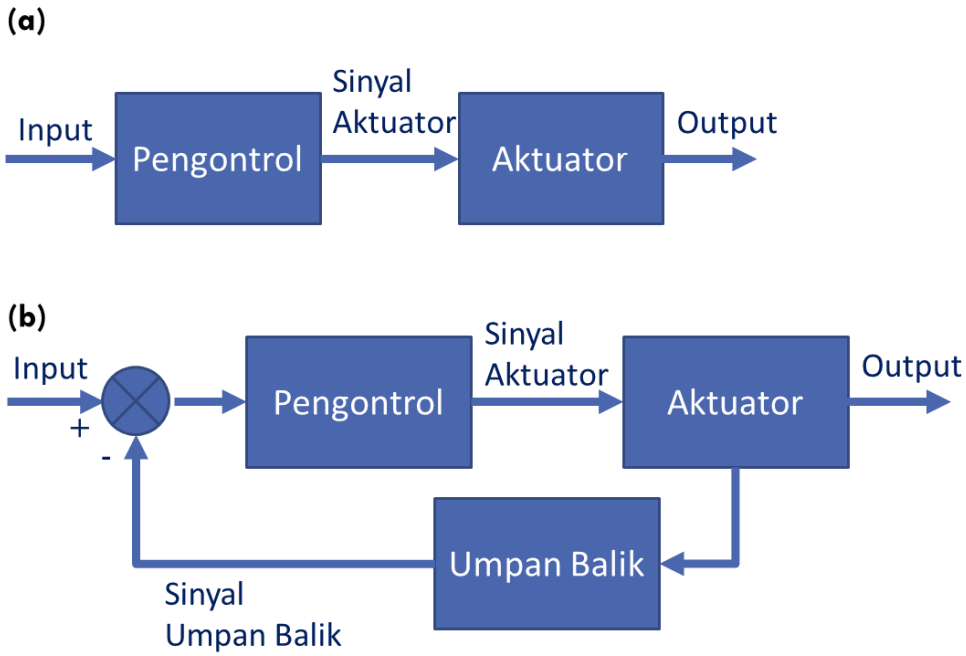
	Institut Teknologi Bandung Program Studi Teknik Fisika		
Praktikum	TF2207 Laboratorium TF II 2 SKS	Dosen	Dr. Ir. Eko Mursito Budi, M.T., IPM Ashari Budi Nugraha, S.T., M.T. Dr.Eng. Muhammad Iqbal, S.T., M.T.

4 REGULATORY CONTROL

4.1 LATAR BELAKANG

Sistem kontrol terdapat dalam banyak sistem atau perangkat, baik sistem atau perangkat yang ada di sekitar kita maupun sistem di plant industri. Secara definisi, sistem kontrol adalah sistem di mana terdapat suatu perangkat yang mengatur, memberikan perintah, atau mengendalikan perangkat lainnya untuk mencapai hasil yang diinginkan. Sebagaimana sebuah sistem, sistem kontrol menerima input dan akan menghasilkan ouput. Jenis sistem kontrol dapat dibedakan berdasarkan klasifikasi sebagai berikut:

- Berdasarkan pada metode analisis dan desain, sistem kontrol dapat dibedakan menjadi sistem linier dan non-linier. Pada sistem kontrol linier, sistem memiliki sifat homogen dan superposisi di mana output dari sistem berbanding lurus dengan inputnya. Sedangkan pada sistem kontrol non-linier, outputnya tidak berbanding lurus dengan inputnya. Pada sistem linier, apabila sinyal input magnitudonya meningkat, maka sinyal ouput juga magnitudonya ikut meningkat dengan tetap mempertahankan bentuk sinyalnya. Sementara itu pada sistem non-linier, bentuk sinyal output dapat berubah seiring dengan berubahnya sinyal input.
- Berdasarkan jenis sinyal, sistem kontrol dapat dibedakan menjadi *continuous time* dan *discrete time*. Pada sistem kontinu, semua variabel sistem merupakan fungsi dari variabel waktu kontinu *t*. Sedangkan pada sistem diskrit, sinyal tidak secara kontinu bervariasi terhadap waktu namun dalam bentuk pulsa-pulsa.
- Berdasarkan jumlah input dan output, sistem kontrol dapat dibedakan menjadi SISO (*single input single output*) dan MIMO (*multiple inputs multiple outputs*).
- Berdasarkan ada tidaknya umpan balik, sistem kontrol dapat dibedakan menjadi sistem *open loop* dan *closed loop*. Pada sistem open loop, tidak terdapat umpan balik terhadap input sistem sehingga aksi kontrol tidak bergantung pada output yang diharapkan. Adapun dalam sistem closed loop, output diumpankan ke input. Dengan demikian, aksi kontrol bergantung pada output yang diharapkan. Perbedaan sistem kontrol open dan closed loop ditampilkan pada **Gambar 4.1**.



Gambar 4.1. Blok diagram sistem kontrol (a) open loop dan (b) closed loop.

Sistem kontrol digunakan dalam banyak aplikasi, seperti pada robotika dan otomasi plant industri. Perangkat aktuator diperlukan dalam suatu sistem kontrol, di mana aktuator berperan sebagai transduser output yang menerjemahkan sinyal kontrol menjadi gerak mekanik, baik gerak translasional maupun gerak rotasional. Aktuator dengan pergerakan translasional antara lain solenoida dan aktuator linear. Adapun aktuator rotasional antara lain meliputi motor DC, motor servo, dan motor stepper. Pada praktikum Lab TF II, akan dipelajari jenis-jenis aktuator rotasional dan cara mengontrolnya untuk tujuan tertentu dengan suatu mikrokontroller. Khusus untuk Modul 4 ini, akan dipelajari pengontrolan dan akuisisi data kecepatan rotasi motor DC.

Akuisisi data adalah proses mengukur besaran fisis menjadi data digital, untuk kemudian disimpan dan diolah lebih lanjut. Secara umum, sistem akuisisi data terdiri dari:

- Sensor
- Pengondisi sinyal analog
- Analog to digital converter
- Pengolah data digital
- Penyimpan data (memori)
- Pengirim data (serial, WiFi, dll.)

Pada praktikum Modul 4 ini akan dilakukan pengontrolan kecepatan motor DC dan akuisisi data kecepatan yang dihasilkan oleh motor menggunakan sensor optocoupler.

4.2 KOMPETENSI

Kompetensi yang akan dikuasai praktikan setelah menyelesaikan praktikum ini adalah:

- Mengimplementasikan penyimpanan data pada memori mikrokontroller EEPROM
- Mengetahui dan dapat mengimplementasi sinyal kontrol output mikrokontroller berbentuk *pulse width modulation* (PWM)

- Memahami cara kerja motor DC dan pengontrolannya menggunakan driver motor L9110
- Mengimplementasikan sensor kecepatan motor DC menggunakan optocoupler
- Mengimplementasikan metode komunikasi secara I2C
- Mengaplikasikan tekik interrupt dan teknik data logging secara online (*ring buffer*)
- Mengaplikasikan pengohalan sinyal menggunakan *moving average*

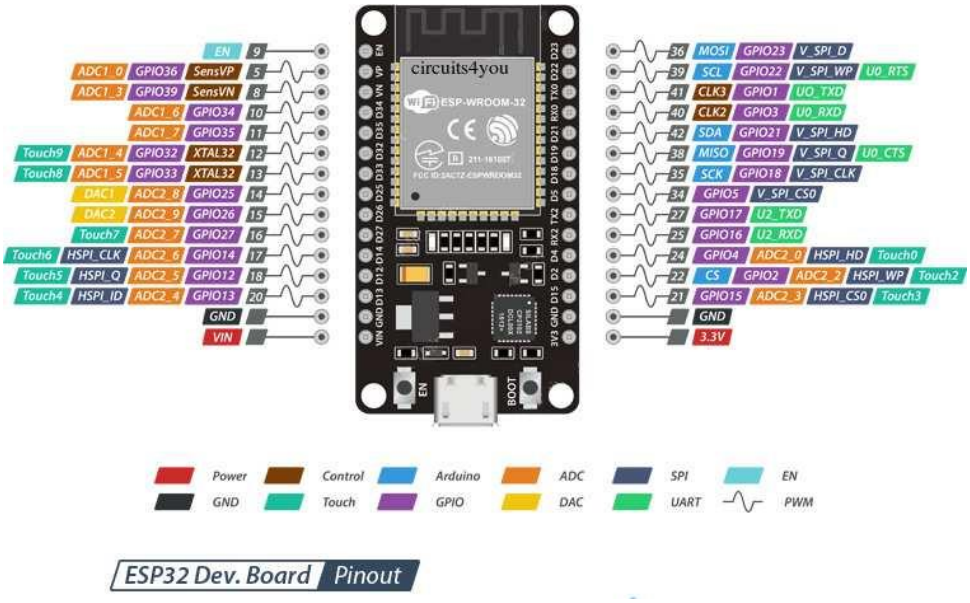
4.3 ALAT & BAHAN

No	Item	Banyak	Keterangan
1	EScope 2022	1	Dari kit
2	Breadboard	1	
3	Kit Motor DC + Optocoupler	1	
4	Driver Motor DC L9110	1	
5	Kabel Daya – USB	1	
5	Kabel Jumper	Secukupnya	
6	Kabel Micro USB	1	Disediakan praktikan
7	Obeng plus kecil	1	
8	Kepala Charger USB (minimal Arus Output 1A)	1	

4.4 PANDUAN TEKNIS

4.4.1 MIKROKONTROLLER

4.4.1.1 PULSE WIDTH MODULATION (PWM)



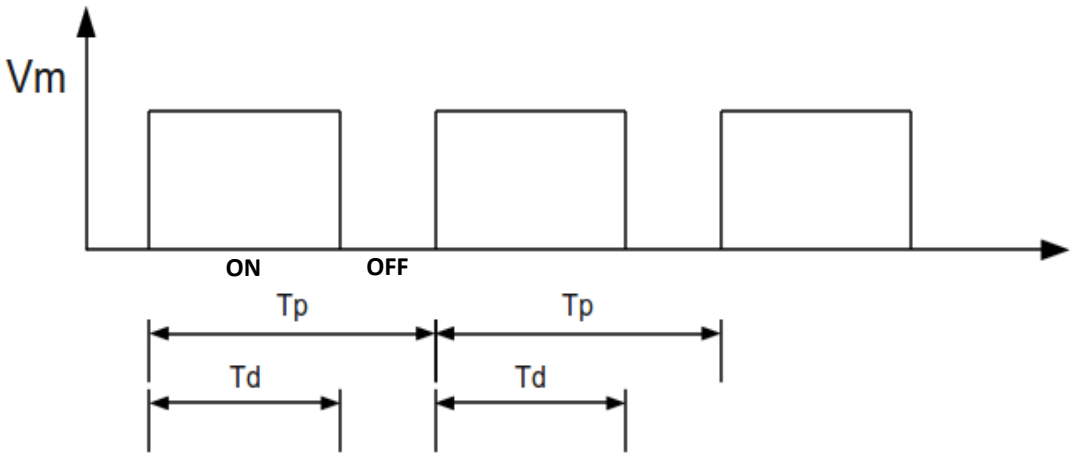
Gambar 4.2. Pengalamatan dan fungsi-fungsi pin pada development board ESP32.

Sebagaimana yang telah dipelajari pada Modul 1 Lab TF 2, *development board* ESP32 memiliki 30 pin yang terekspos dan masing-masing dapat digunakan untuk fungsi yang beragam (**Gambar 4.2**). Sebagai contoh, pin-

pin pada ESP32 dapat difungsikan sebagai pin input *analog-to-digital converter* (ADC) yang berfungsi mencacah sinyal bentuk analog menjadi digital. Salah satu fungsi pin lain yang akan digunakan pada Modul 4 ini adalah fungsi untuk membangkitkan sinyal output *pulse width modulation* (PWM).

Semua pin GPIO pada ESP32 dapat difungsikan untuk mengeluarkan sinyal PWM (perhatikan simbol PWM di **Gambar 4.2**). Lebih lanjut, kita bisa mengeluarkan sinyal PWM dari 16 channel independen (channel 0 hingga channel 15) yang dapat dikonfigurasi dengan properties yang berbeda-beda. PWM sendiri merupakan teknik modulasi dengan mengubah lebar pulsa sesuai dengan angka digital tertentu, dengan nilai frekuensi dan amplitudo yang tetap. PWM dapat dianggap lawan dari ADC, yang mana suatu mikrokontroller menghasilkan/membangkitkan output sinyal analog. Pada praktikum Modul 4 ini, sinyal PWM akan digunakan untuk mengendalikan nyala gelap-terap lampu LED dan motor DC.

Sesuai dengan namanya, sinyal PWM berbentuk sinyal pulsa, di mana amplitudo dan perioda satu siklus penuhnya tetap namun lebar pulsa ON-nya dapat diubah-ubah berdasarkan nilai digital tertentu. Bentuk sinyal PWM ditampilkan pada **Gambar 4.3**. ESP32 memiliki resolusi 1 hingga 16 bit PWM, artinya angka digital yang bisa dipilih untuk mengatur lebar pulsa PWM adalah 0 – 65535. Adapun resolusi standar yang biasa digunakan adalah 8 bit (angka digital 0 – 255). Lebar pulsa yang diatur dengan angka digital yang diberikan pada pin PWM akan menentukan tingkat tegangan keluaran rata-rata yang dihasilkan. Lebar pulsa (durasi lamanya sinyal ON) dalam satu siklus sinyal disebut dengan istilah *duty cycle*.

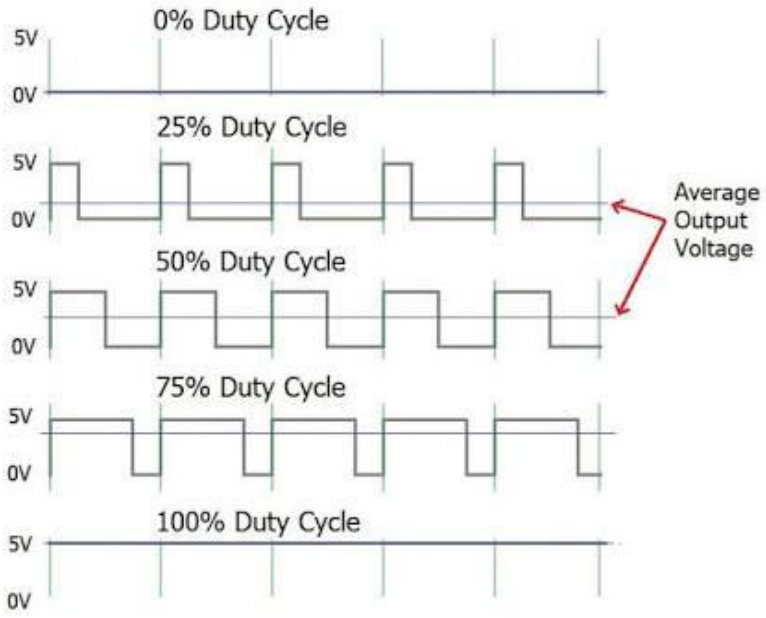


Gambar 4.3. Bentuk sinyal pulsa PWM. Tp adalah periode penuh satu sinyal PWM, sedangkan Td adalah periode duty cycle (durasi lamanya sinyal ON).

Periode *duty cycle* dapat dihitung dengan persamaan berikut ini,

$$T_d = \frac{D}{(2^n - 1)} T_p$$

di mana Td adalah periode *duty cycle*, *D* adalah angka digital yang diatur/dikeluarkan, *n* adalah resolusi PWM, dan Tp adalah periode satu siklus penuh sinyal PWM (durasi lama sinyal ON + sinyal OFF). Besaran d*uty cycle* juga pada umumnya dituliskan dalam bentuk persen, di mana persentase *duty cycle* sekali lagi akan menentukan tegangan keluaran rata-rata yang dibangkitkan oleh pin PWM (**Gambar 4.4**). Hubungan matematis antara *duty cycle* dengan tegangan keluaran rata-rata adalah $\mathrm{V_{out}} = \mathrm{Duty\ Cycle} \times \mathrm{V_{in}}$.



Gambar 4.4. Hubungan persentase duty cycle dengan tegangan keluaran rata-rata yang dibangkitkan oleh pin PWM pada mikrokontroller.

Berikut ini adalah tahap-tahap umum dalam pemrograman mikrokontroller untuk membangkitkan sinyal PWM:

1. Memilih channel PWM, terdapat channel 0 hingga 15.
2. Mengatur frekuensi sinyal PWM (hingga 10 KHz).
3. Menentukan resolusi sinyal *duty cycle*, pada umumnya 8 bit (0 – 255).
4. Menentukan pin GPIO di mana sinyal output PWM akan dibangkitkan.

Membuat rutin program untuk membangkitkan sinyal PWM dengan menyertakan channel serta *duty cycle*-nya.

4.4.1.2 MEMORI PENYIMPANAN DATA

Memori adalah bagian mikrokontroller yang berfungsi untuk menyimpan data. Secara umum, terdapat tiga jenis memori pada mikrokontroller, yaitu:

- **SRAM** (static random-access memory), memori kerja tempat program membuat dan memanipulasi variabel saat berjalan; bersifat *volatile*
- **ROM (**Read only memory), tempat program inti (*firmware*) disimpan; bersifat non-volatile
- **Flash memory**, tempat program disimpan; bersifat *non-volatile*
- **EEPROM (***Electrically Erasable Programmable Read-Only Memory*), yaitu ruang memori di mana program dapat menyimpan informasi jangka panjang; bersifat *non-volatile*

Untuk ESP32, memori yang ada adalah:

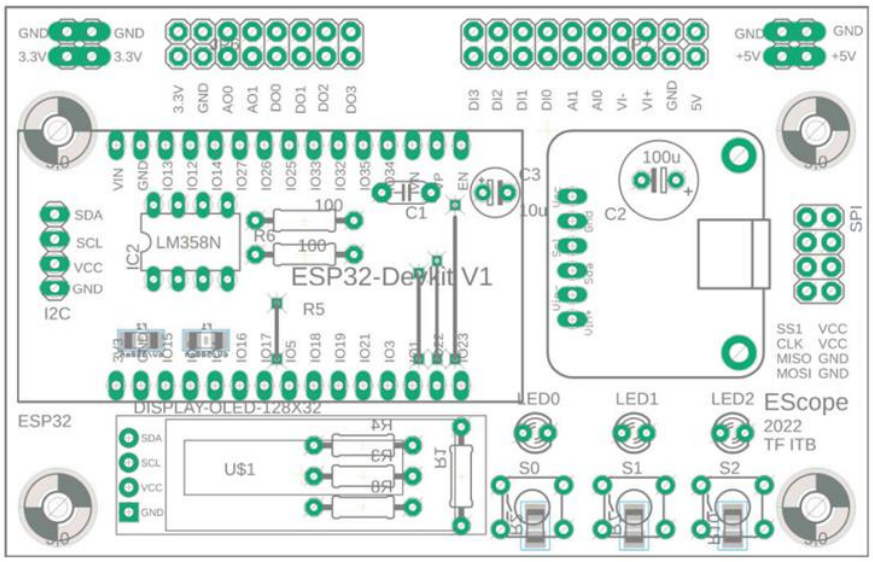
- SRAM sebanyak 520 KB
- ROM sebanyak 448 KB
- Flash Memory sebanyak 4 MB (maksimum 4 KB difungsikan sebagai EPPROM)

ESP32 tidak memilik EEPROM. Karena itu untuk menyimpan data, digunakan sebagian flash memory dialokasikan untuk penyimpan data, maksimum sebanyak 4 KB. Hal ini menjadi batasan pada banyak data yang dapat disimpan selama akuisisi data.

4.4.2 PERANGKAT KERAS

4.4.2.1 ESCOPE

EScope adalah board pengembangan berbasis ESP32 yang dilengkapi dengan beberapa piranti standar, dan didesain agar mudah dikoneksikan ke breadboard.



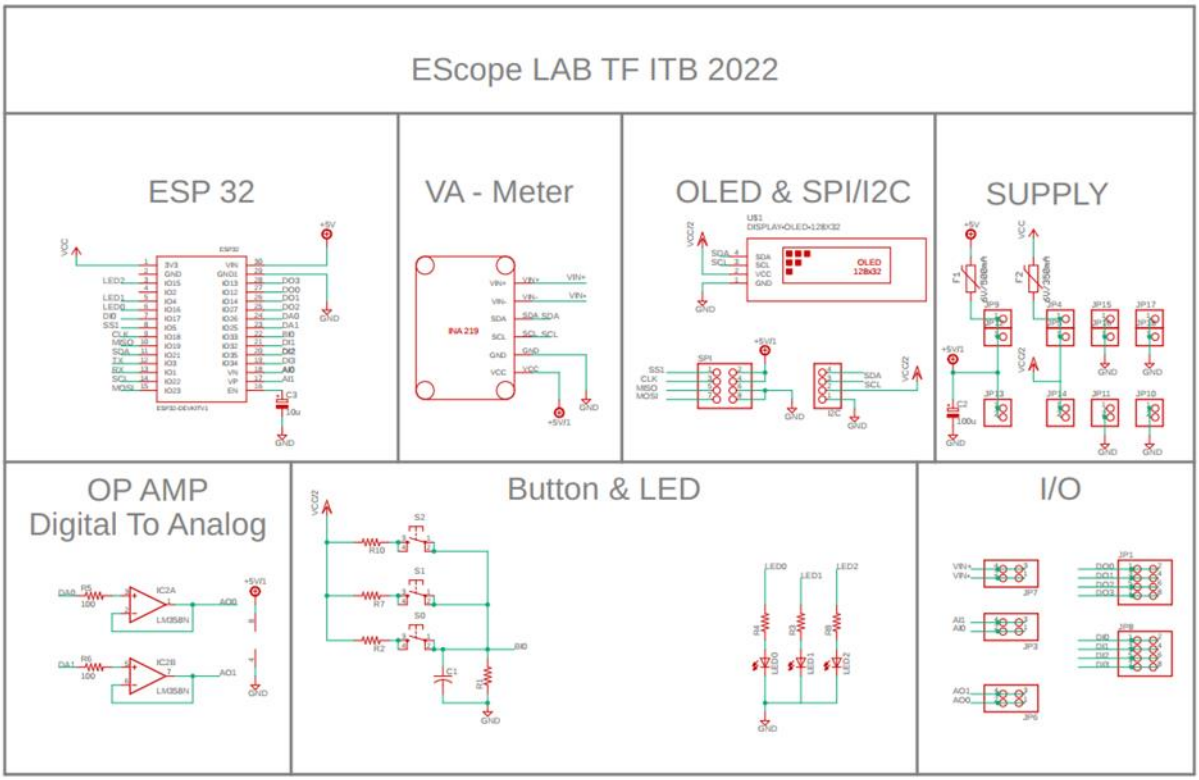
Gambar 4.5. Board ESCOPE

Komponen standar pada ESCOPE meliputi:

- Konekt
- OLED resolusi 128x32 yang terkoneksi melalui I2C
- INA219 yang juga terkoneksi melalui I2C
- Tiga buat tombol, terkoneksi secara pull-down ke GPIO 13
- Tiga LED (Merah, Kuning, dan Hijau), terkoneksi secara source ke pin GPIO 16, 4, dan 15
- Dua Op-amp sebagai keluaran analog, terkoneksi ke DA pin GPIO 26 dan 25

Selanjutnya terdapat soket AO, DO, AI, dan DI untuk sambungan kabel ke breadboard.

Detail skema ESCOPE adalah sebagai berikut.



Gambar 4.6. Skema EScope

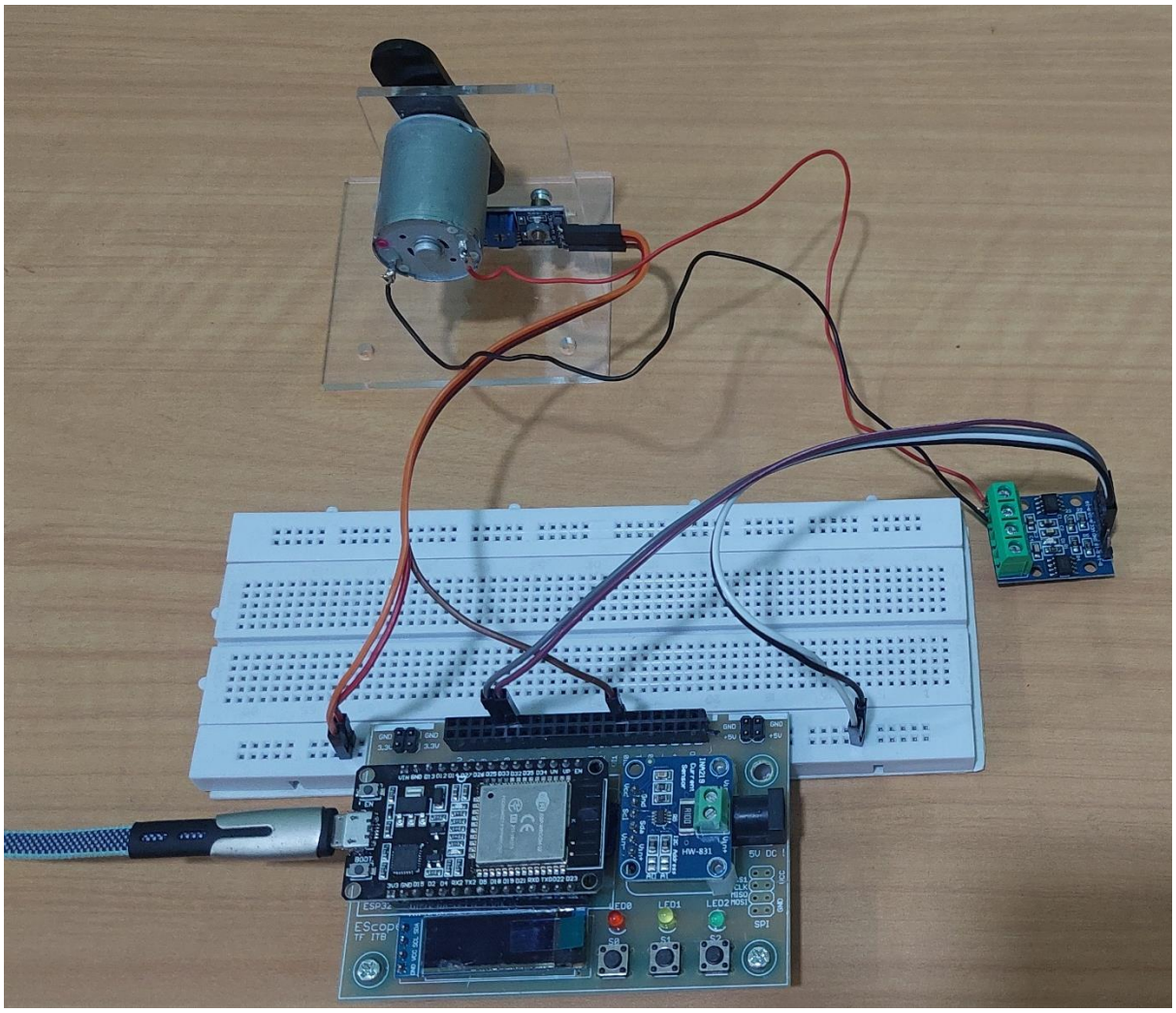
4.4.2.2 BREADBOARD

Skema rangkaian dan foto komponen yang perlu dirangkai pada breadboard adalah sebagai berikut (**Gambar 4.7**).

Pada **Tabel 4.1** di bawah ini adalah wiring (pengkabelan) menggunakan kabel jumper *male-to-male* yang perlu disiapkan sesuai dengan **Gambar 4.7**.

Tabel 4.1. Wiring (pengkabelan) motor DC, driver, optocoupler, dan EScope.

Motor DC	Driver L9110	Opto-Coupler	EScope	Breadboard
Kabel merah dan hitam	Soket Motor A	-	-	-
-	GND			Line GND (kanan)
-	VCC			Line +5V
-	A-1A		DO0	-
-	A-1B		DO1	-
-	-	GND		Line GND (kiri)
-	-	VCC		Line +3.3V
-	-	D0	DI0	-



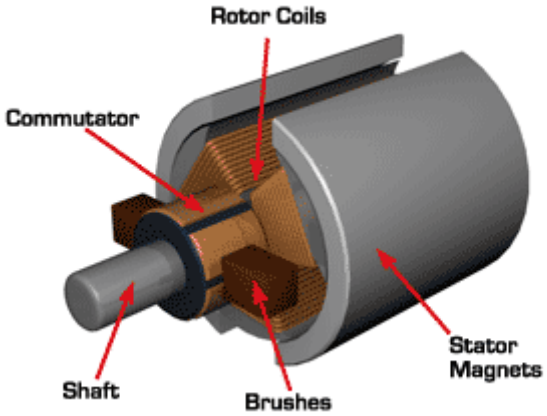
Gambar 4.7. Wiring motor DC, driver, dan breadboard.

PERHATIAN!

Tambahan: Sambungkan kabel daya-USB pada jack power EScope ke kepala charger 5VDC (minimal 1A). Kemudian colokkan kepala chargernya ke colokan listrik PLN (220V) di soket listrik yang tersedia di laboratorium.

4.4.2.3 MOTOR DC

Motor dc magnet permanen (*permanent magnet dc motor*/PMDC) dapat menghasilkan gerak putar/rotasi yang kontinu dengan sinyal kendali yang juga kontinu, sehingga dikategorikan sebagai *continuous-drive actuator*. Motor dc terdiri dari stator yang berbahan magnet permanen, komutator, *brushes*, sumbu, dan kumparan kawat rotor (**Gambar 4.8)**.

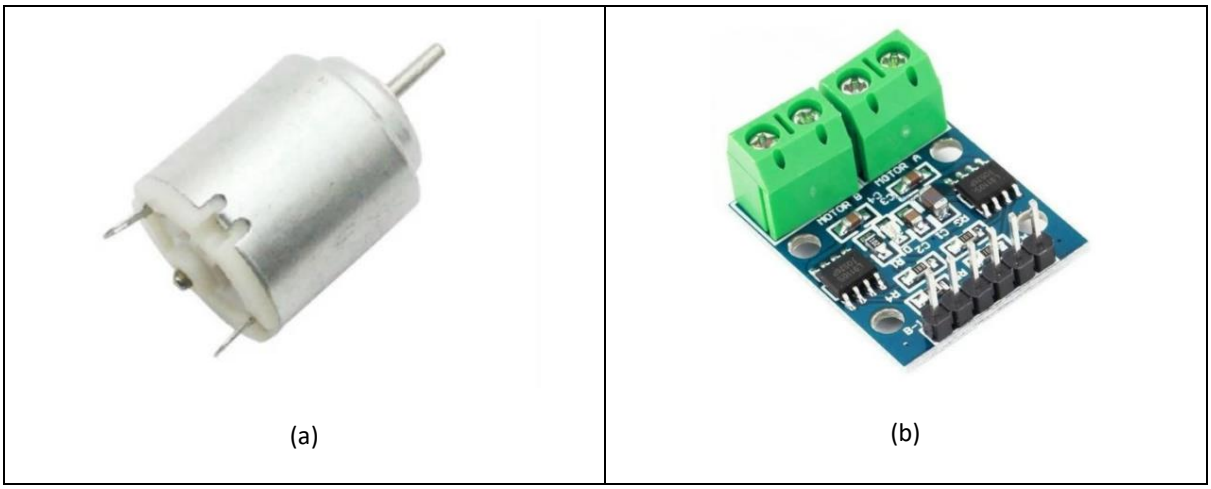


Gambar 4.8. Konstruksi motor DC magnet permanen.

Motor DC banyak digunakan dalam perangkat-perangkat listrik, seperti:

- Vibrator ponsel
- Peralatan industri dan rumah tangga
- Disk driver
- Bor listrik DC
- Kipas DC
- Peralatan elektronik lainnya

Pada bidang robotika, motor DC banyak digunakan. Rating tegangan motor DC bermacam-macam, namun yang akan kita gunakan pada praktikum Modul 4 ini adalah motor DC 5V (**Gambar 4.9a**). Arah putaran motor DC (searah atau berlawanan arah jarum jam) dapat diatur dengan membalik polaritas tegangan yang masuk ke terminal daya motor DC. Untuk memudahkan pengendalian arah putar motor DC, digunakan driver motor berbasis rangkaian H-Bridge. Pada praktikum Modul 4 ini, akan digunakan driver motor H-Bridge L9110 (**Gambar 4.9b**). Driver ini dapat mengendalikan dua buah motor DC secara independen, namun kali ini yang akan kita manfaatkan hanya untuk mengendalikan satu buah motor DC saja.



(a) (b)

Gambar 4.9. (a) Motor dc 5V dan (b) driver motor DC H-Bridge L9110.

Pemrograman sederhana untuk mengendalikan gerak satu buah motor DC menggunakan driver L9110 tersebut ditampilkan pada **Kode 4.1**. Perhatikan bahwa untuk mengendalikan motor DC, kita akan pergunakan kembali sinyal PWM dari mikrokontroller.

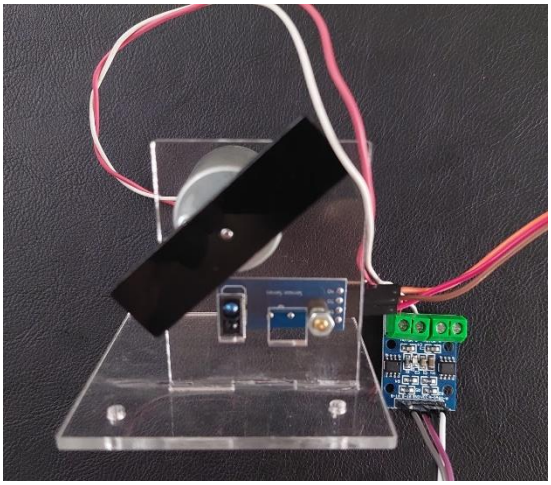
```
/* Program : Motor DC
 * Test Motor DC
 */

// konstanta untuk PWM motor
#define PWM_CHANNEL 0
#define PWM_RES   8

#define PWM_MIN -255
#define PWM_MAX 255

// Memilih pin PWM untuk input driver motor
int pin_motorA = 13; // pin GPIO 13 alias DO0
int pin_motorB = 12; // pin GPIO 12 alias DO1

// Mengatur properties PWM
int motor_freq = 15;
int motor_pwm = 0;
```

```
void setup() {
  // Mengatur pin sebagai output
  pinMode(pin_motorA, OUTPUT);
  pinMode(pin_motorB, OUTPUT);

  // Memasangkan channel ke GPIO yang akan digunakan
  ledcAttachPin(pin_motorA, PWM_CHANNEL);

  // Konfigurasi fungsi PWM
  ledcSetup(PWM_CHANNEL, motor_freq, PWM_RES);
}

/* Mengatur kecepatan dan arah putaran motor DC
 * pwm = 0 : berhenti
 * pwm 1-255 : putar kanan
 * pwm -(1-255) : putar kiri
 */
void dcMotorGo() {
  if (motor_pwm >= 0) {
    ledcWrite(PWM_CHANNEL, motor_pwm);
    digitalWrite(pin_motorB, LOW);
  }
  else {
    ledcWrite(PWM_CHANNEL, 255+motor_pwm);
    digitalWrite(pin_motorB, HIGH);
  }
}

void loop(){
  // motor DC berputar ke kanan
  for(motor_pwm = 0; motor_pwm <= PWM_MAX; motor_pwm++){
    dcMotorGo();
    delay(15);
  }
  for(motor_pwm = 255; motor_pwm >= 0; motor_pwm--){
    dcMotorGo();
    delay(15);
  }
  delay(3000);

  // motor DC berputar ke kiri
  for(motor_pwm = 0; motor_pwm >= PWM_MIN; motor_pwm--){
    dcMotorGo();
    delay(15);
  }
  for(motor_pwm = PWM_MIN; motor_pwm <= 0; motor_pwm++){
    dcMotorGo();
    delay(15);
  }
  delay(3000);
}
```

Kode 4.1. Kode sumber sederhana untuk mengendalikan motor DC menggunakan driver motor L9110.

Jika kode sumber di atas dijalankan, motor DC akan bergerak dari keadaan berhenti, berputar ke arah kanan, kemudian berhenti kembali. Lalu berputar ke arah kiri dan kembali diam. Sebagaimana telah dijelaskan sebelumnya, arah putaran motor DC dapat diatur dengan menukar polaritas tegangan yang masuk ke terminal daya motor.

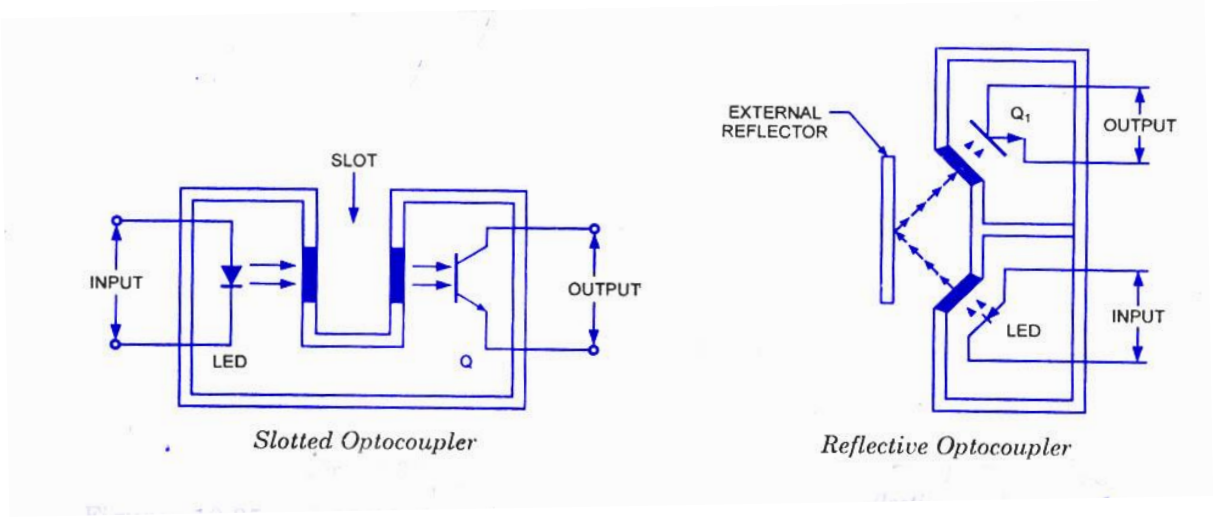
Pada Modul 4 ini, motor DC dilengkapi dengan dudukan dan bilah yang terbuat dari akrilik (**Gambar 4.10**).



Gambar 4.10. Dudukan dan bilah motor DC yang terbuat dari akrilik.

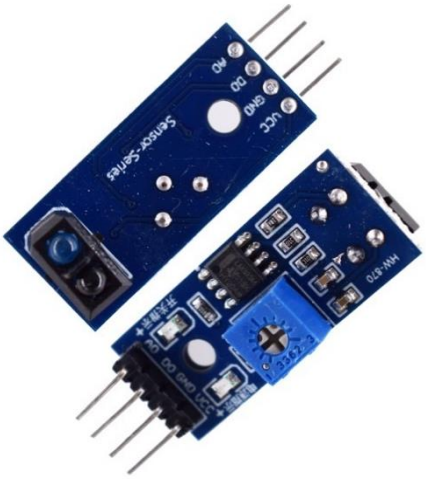
4.4.2.4 SENSOR OPTO-COUPLER

Kita dapat mengimplementasi pengukuran kecepatan rotasi sebuah motor DC dengan memanfaatkan modul opto-coupler. Modul opto-coupler terdiri dari sepasang transmitter dan sensor cahaya. Cahaya yang ditransmisikan oleh transmitter (LED) dapat berupa cahaya tampak maupun sinar inframerah. Sementara itu, sensor cahaya merupakan komponen yang sensitif terhadap cahaya, seperti phototransistor atau photodioda. Cara kerja opto-coupler diilustrasikan pada **Gambar 4.11**. Kegunaan opto-coupler selain sebagai pengukur kecetapan rotasi motor DC adalah sebagai saklar/switch elektronik yang dapat berfungsi secara otomatis. Terdapat dua konfigurasi pasangan transmitter dan sensor cahaya, yaitu saling berhadapan (*slotted optocoupler*) dan saling besebelahan (*reflective optocoupler*). Modul opto-coupler yang akan kita gunakan di Modul 4 ini adalah konfigurasi yang kedua, yaitu transmitter dan sensor cahaya ditempatkan saling berdampingan, namun masing-masing terisolasi secara terpisah. Dengan demikian, sensor cahaya akan menghasilkan keadaan HIGH apabila cahaya yang dikeluarkan oleh transmitter dipantulkan oleh suatu benda yang dapat memantulkan cahaya tersebut. Dalam hal ini, benda yang berperan untuk memantulkan cahaya dari transmitter adalah bilah/baling-baling motor DC.



Gambar 4.11. Ilustrasi skematik dua jenis opto-coupler (slotted dan reflective).

Modul opto-coupler yang akan digunakan pada praktikum Modul 4 ini adalah opto-coupler HW-870 (**Gambar 4.12**). Modul ini memiliki jarak pantulan deteksi antara 1 mm – 25 mm. Tegangan operasinya adalah 3,3V – 5 V. Kemudian bentuk keluarannya berupa output switch digital (0 dan 1).



Gambar 4.12. Modul opto-coupler HW-870.

Kode sumber di bawah ini **(Kode 4.2**) merupakan contoh program sederhana untuk melakukan penghitungan (*counter*) pulsa sensor opto-coupler menggunakan teknik interrupt ketika sensor cahaya menerima pantulan sinar inframerah dari transmitter yang dipantulkan oleh bilah/baling-baling pada motor DC.

```
/* Program : Opto Coupler
 * Test Optocoupler Sensor
 * Dengan memakai interrupt untuk counter
 *
 * Ref: https://www.theengineeringprojects.com/2021/12/esp32-
interrupts.html
 *
 * (c) Eko M. Budi, 2022
 */
```

```
#include <TFScope.h>

#define DI_OPTO DI0

volatile unsigned opto_counter = 0;
portMUX_TYPE opto_mux = portMUX_INITIALIZER_UNLOCKED;

void optoISR() {
  portENTER_CRITICAL_ISR(&opto_mux);
  opto_counter++;
  portEXIT_CRITICAL_ISR(&opto_mux);
}

unsigned optoCount() {
  unsigned count;
  portENTER_CRITICAL_ISR(&opto_mux);
  count = opto_counter;
  portEXIT_CRITICAL_ISR(&opto_mux);
  return count;
}

unsigned optoReset() {
  unsigned count;
  portENTER_CRITICAL_ISR(&opto_mux);
  count = opto_counter;
  opto_counter=0;
  portEXIT_CRITICAL_ISR(&opto_mux);
  return count;
}

void optoStart() {
  opto_counter=0;
  pinMode(DI_OPTO, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(DI_OPTO), optoISR, FALLING);
}

void optoStop() {
  detachInterrupt(digitalPinToInterrupt(DI_OPTO));
}

void plotSignal() {
  bool d = digitalRead(DI_OPTO);
  Serial.println(d);
}

void plotHeader() {
  Serial.print("Pulse Counter=");
  Serial.println(optoCount());
}

void setup() {
  Serial.begin(500000);
  delay(1000);
  Serial.println();
  Serial.println("OptoCoupler Test");
  optoStart();
}

void loop() {
  plotHeader();
```

```
  plotSignal();
  delay(10);
}
```

Kode 4.2. Kode program untuk menghitung banyaknya pantulan sinar inframerah yang diterima oleh sensor opto-coupler menggunakan teknik interrupt.

Pada contoh kode program ini, gerak bilah motor DC digerakkan secara manual oleh tangan. Penghitungan / *counter* bilah motor DC secara otomatis akan dilakukan saat praktikum untuk menghitung kecepatan putar motor DC.

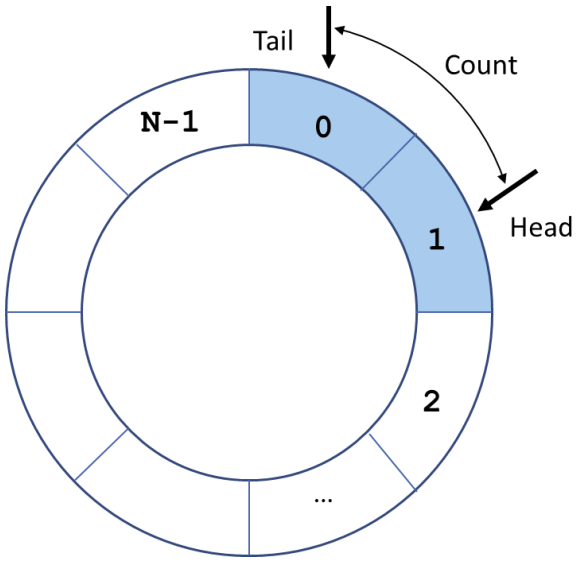
4.4.3 PEMROGRAMAN DASAR

4.4.3.1 RING BUFFER

Pada akuisisi data secara on-line, mikroprosesor melakukan pengukuran, lalu data diolah dan dikirimkan ke komputer secara terus menerus. Dengan demikian data perlu ditampung sementara secara berkelanjutan.

Untuk itu diperlukan buffer data yang tak perlu terlalu panjang, namun harus bisa berjalan terus menerus. Struktur data yang efisien digunakan untuk ini adalah RingBuffer (**Gambar 4.13**). Prinsipnya adalah:

- Buffer berupa array dengan ukuran NDATA
- Array diisi sesuai index head yang dinaikkan satu sebelum mulai mengisi. Jika head sudah mencapai NDATA, maka diputar kembali ke 0. Namun jika saat akan mengisi ternyata head menjadi sama dengan tail, maka artinya buffer sudah penuh. Untuk akuisisi data, kebijakan yang diambil adalah tindis saja data yang lama. Kalau untuk operasi lain, biasanya data baru yang akan diabaikan.
- Array lalu dibaca sesuai index tail. Jika tail masih sama dengan head, tandanya buffer masih kosong dan jangan baca-apa apa. Kalau sudah selesai baca, maka tail dinaikkan 1, dan bila sudah mencapai NDATA, maka putar kembali ke 0.



Gambar 4.13. Ilustrasi ring buffer.

Untuk memudahkan pemrograman, telah disediakan pustaka TFRingBuffer. Kelas ini dapat menyimpan item berupa tipe data apa saja, termasuk array dan struct. Contoh penggunaannya adalah sebagai berikut:

```
#include <TFRingBuffer.h>

// konstanta banyak item ring
#define N_DATA 10

// definisikan tipe data yang akan disimpan
#define T_DATA int

// Pesan kelas RingBuffer
// Ini adalah Teknik pemrograman baru yang disebut template
// Ikuti saja contohnya
RingBuffer<N_DATA, T_DATA> rbuff;

int signal;

// di fungsi bisa dipanggil
void any_function() {
  rbuff.reset(); // mengosongkan

  rbuff.put(signal);   // menaruh satu item ke akhir buffer

  rbuff.take(signal);  // mengambil satu item dari awal buffer (FIFO)
  rbuff.pop(signal);  // mengambil satu item dari akhir buffer
(LIFO)
}
```

Beberapa fungsi yang dapat digunakan pada RingBuffer adalah sebagai berikut:

Fungsi kelas LinierBuffer	Deskripsi
reset()	Mengosongkan buffer
void put(TYPE &data)	Menaruh data ke akhir buffer (head akan maju)
bool take(TYPE &data)	Mengambil data dari awal buffer (tail akan maju)
bool pop(TYPE &data)	Mengambil data dari akhir buffer (head akan mundur)
bool isFull()	Memeriksa buffer sudah penuh (jika put, akan menghapus data pertama)
bool isEmpty()	Memeriksa apakah buffer kosong (tak bisa melakukan take / pop)
int count()	Melihat banyaknya item di buffer

4.4.3.2 FLASHBUFFER

FlashBuffer adalah pustaka sediaan (TFFlashBuffer), berupa ring buffer yang tersimpan di EEPROM (flash memory pada ESP32). Pustaka inilah yang dapat dimanfaatkan untuk menyimpan data secara permanen. Contoh penggunaan dasarnya adalah sebagai berikut:

```
#include <TFRingBuffer.h>

// konstanta banyak item ring
// Harus tak melebihi ukuran Flash Memory yang tersedia
#define N_DATA 10

// Alamat untuk meletakkan buffer
#define ADDR_BUFFER 0

// definisikan tipe data yang akan disimpan
#define T_DATA int

// Pesan kelas FlashBuffer
FlashBuffer<N_DATA, T_DATA> rbuff(ADDR_BUFFER);

int signal;

// di fungsi bisa dipanggil
void anyFunction() {
  rbuff.reset(); // mengosongkan

  rbuff.put(signal);   // menaruh satu item ke akhir buffer

  rbuff.take(signal);  // mengambil satu item dari awal buffer (FIFO)

  rbuff.pop(signal);  // mengambil satu item dari akhir buffer
(LIFO) }
```

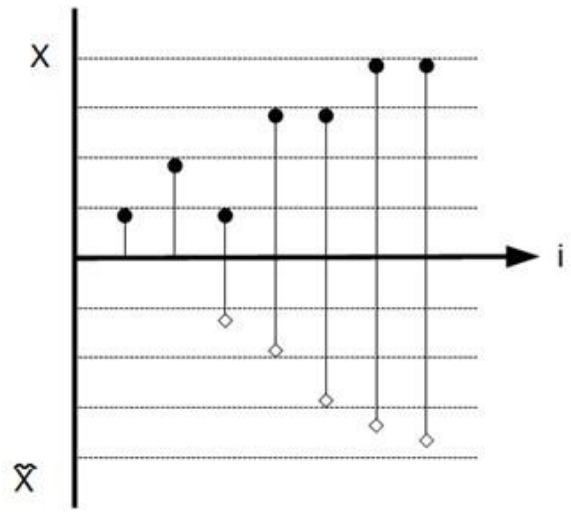
Beberapa fungsi yang dapat digunakan pada FlashBuffer adalah sebagai berikut:

Fungsi kelas LinierBuffer	Deskripsi
unsigned memSize()	Menggembalikan ukuran memory yang diperlukan pada EEPROM
void init()	Menyiapkan buffer di EPPROM (dari kosong)
void erase()	Menghapus buffer dari EEPROM (tak akan bisa diload lagi)
bool load()	Memuat buffer dari EPPROM (harus sudah di-init)
void put(TYPE &data)	Menaruh data ke akhir buffer (head akan maju)
bool take(TYPE &data)	Mengambil data dari awal buffer (tail akan maju)
bool pop(TYPE &data)	Mengambil data dari akhir buffer (head akan mundur)
bool isFull()	Memeriksa buffer sudah penuh (jika put, akan menghapus data pertama)

bool isEmpty()	Memeriksa apakah buffer kosong (tak bisa melakukan take / pop)
unsigned count()	Melihat banyaknya item di buffer
unsigned row()	Melihat Panjang buffer (banyak item yang bisa disimpan)
unsigned written()	Melihat banyaknya data yang sudah pernah disimpan ke buffer ini
void getFirst(TYPE &d)	Melihat data pertama (tail), tanpa mengambil
void getFirst(int i, TYPE &d)	Melihat data pertama ke-I (tail+i), tanpa mengambil
void getLast(TYPE &d)	Melihat data terakhir (head), tanpa mengambil
void getLast(int i, TYPE &d)	Melihat data terakhir ke-I (head-i), tanpa mengambil

4.4.3.3 MOVING AVERAGE

Salah satu masalah dalam pengukuran adalah noise (derau). Akibatnya, grafik hasil akuisisi data tidak akan mulus. Untuk mengatasi derau, sebaiknya data ditapis (filter) dulu sebelum dikirimkan. Metode penapisan lanjut kelak bisa Anda pelajari di kuliah Pengolahan Sinyal. Untuk praktikum ini, kita akan pakai teknik sederhana yang disebut rata-rata bergerak (*moving average*). Prinsipnya adalah, pada waktu pencuplikan ke-*i*, maka data yang dikirimkan adalah rata-rata dari *N* data pengukuran terakhir. **Gambar 4.** 14 berikut mengilustrasikan proses moving average dengan *N* = 3. Perhatikan bahwa data sebenarnya (X) mulai diukur pada *i* = 1. Setelah pengukuran ke *N*, barulah dihasilkan data rata-rata bergerak (*X*). Nampak bahwa sinyal asli (*X*) yang lebih banyak deraunya bisa menjadi lebih mulus setelah dirata-ratakan (*X*).



Gambar 4. 14. Ilustrasi moving average.

Kalkulasi moving average pada waktu tertentu (t) ini bisa dirumuskan sebagai:

$\overline{x(t)} = \frac{1}{N}\sum_{i=0}^{N} x(t-i)$	*x:* data *t:* waktu sekarang (sampling terakhir) *i*: sampling sebelumnya *N* : panjang *window moving average*

Implementasi kalkulasi ini bisa dilakukan pada suatu ring buffer, yaitu dengan mengambil data terakhir sebanyak N (last(0) hingga last(n)), dijumlahkan dan kemudian dibagi dengan N. Meski demikian, perlakuan khusus diperlukan jika data yang terukur belum mencapai *N*, dalam hal ini cukup rata-ratakan sebanyak data yang sudah ada.

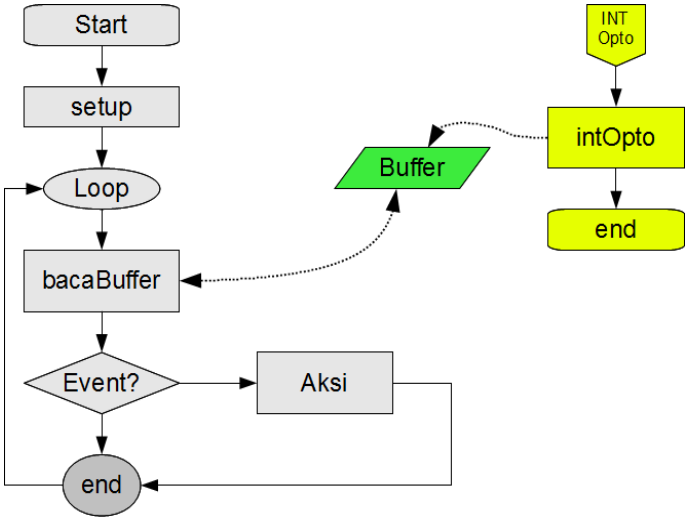
4.4.3.4 TEKNIK INTERRUPT

Interupsi adalah fitur yang memungkinkan mikrokontroller untuk :

- Mendeteksi adanya picuan (trigger) pada masukan digital.
- Jika picuan muncul, seketika itu juga aliran program normal akan dihentikan.
- Mikrokontroller memanggil suatu rutin kecil, *interrupt service routine*, untuk berjalan. Rutin ini harus menyelesaikan tugas secepat-cepatnya, lalu *Return*.
- Program normal akan dilanjutkan kembali.

Pada praktikum Modul 4 ini, teknik interrupet akan digunakan untuk menerima sinyal dari perangkat input berupa sensor optocoupler. Menerima sinyal dari perangkat input (misalnya tombol atau optocoupler) dengan interupsi akan jauh lebih andal dibandingkan membacanya memakai digitalRead biasa. Perhatikan beberapa hal berikut:

- ESP32 bisa menerima interupsi semua pin GPIO.
- Ada tiga jenis picuan yakni RISING (berubah dari LOW ke HIGH), FALLING (berubah dari HIGH ke LOW), dan CHANGE (berubah kapan saja ketika nilai pin berubah dari LOW ke HIGH atau HIGH ke LOW).
- Rutin interupsi harus dibuat sebagai suatu fungsi void tersendiri.
- Jika rutin interupsi perlu mengakses variabel global, maka variabel tersebut harus dideklarasikan sebagai *volatile*.
- Agar rutin interupsi siap beraksi, maka setup()harus memasangnya dulu dengan fungsi attachInterrupt.
- Jika di bagian rutin normal (loop dan semua fungsi yang terpanggil oleh loop) akan mengakses variabel global, maka harus dipastikan bahwa saat itu interupsi tidak boleh terjadi.



Gambar 4.15. Algoritma foreground dan background.

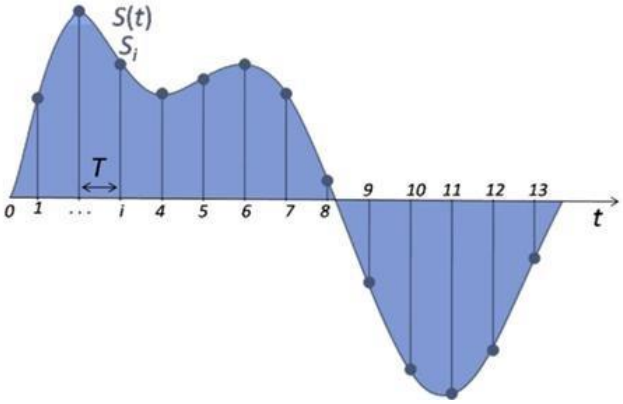
Untuk memrogram interupsi, biasanya dipakai algoritma *foreground-background* seperti ditunjukkan pada **Gambar 4.15**. Penjelasan algoritma tersebut adalah sebagai berikut,

- Rutin normal program (setup dan loop) berjalan sebagai *background*, yaitu proses prioritas rendah yang bisa di-interupsi kapan saja. Seperti biasa, rutin ini disarankan memakai algoritma *event-driven* (jalankan aksi bisa ada event).
- Rutin interupsi (dalam hal ini INT Opto) berjalan sebagai *foreground*, yaitu proses prioritas tinggi yang akan dijalankan kapan saja jika ada picuan interupsi.
- Antara *foreground* dan *background*, ada suatu buffer untuk saling bertukar data.

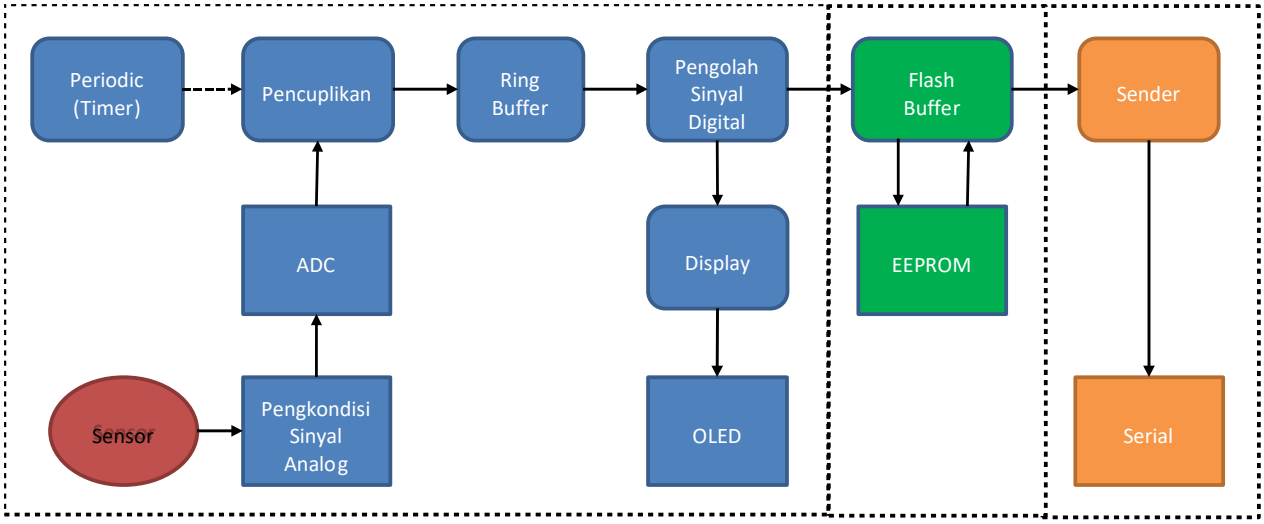
4.4.4 APLIKASI

4.4.4.1 AKUISISI DATA

Pada praktikum ini, kita akan membangun suatu sistem akuisisi data berbasis mikrokontroller. Suatu mikrokontroller, tidak sanggup mengukur sinyal secara kontinu terus-menerus. Yang dapat dilakukan oleh mikrokontroller sebagai pengakuisisi data adalah melakukan pengukuran berkali-kali secara periodik. Pengukuran berkali-kali secara periodik dikenal dengan pencuplikan data (data sampling). Gambar berikut memperlihatkan contoh sinyal kontinyu S(t) yang telah dicuplik dengan periode sampling ts = T. Terlihat bahwa hasilnya adalah sinyal waktu diskrit (Si), digambarkan sebagai garis-garis vertikal dengan bulatan. Sinyal waktu diskrit adalah sinyal hanya mengandung informasi pada waktu-waktu tertentu saja (dinyatakan oleh indeks i). Dengan kata lain, pencuplikan akan menyebabkan adanya informasi yang hilang dari segi waktu. Untuk mengurangi informasi yang hilang, maka periode cuplik (T) perlu diperkecil seminimal mungkin. Namun seperti akan kita lihat nanti, ada beberapa hal yang membatasi hal ini.



Gambar 4.16. Sinyal digital pada akuisisi data



Gambar 4.17. Blok diagram sistem akuisisi data menggunakan mikrokontroller.

Diagram blok umum sistem akuisisi data dengan pengontrol mikro ditunjukkan pada **Gambar 4.17** di atas. Bagian pertama (kita sebut Meter) berfungsi sebagai pengukur dan penampil data online yang terdiri atas:

- Sensor: adalah komponen yang berfungsi mengubah sinyal fisis menjadi sinyal listrik. Ada berbagai jenis sensor yang harus dipilih berdasarkan kondisi sinyal listrik yang akan diukur.
- Pengkondisi sinyal analog: berupa rangkaian elektronik yang berfungsi mengubah sinyal dari sensor menjadi sinyal yang sesuai dengan spesifikasi ADC (biasanya tegangan 0-5 V).
- Konverter Analog Ke Digital (ADC): adalah komponen untuk mengubah tegangan menjadi data biner yang dapat dibaca oleh mikrokontroller. Pada praktikum ini digunakan ADC internal ESP32 maupun INA-219.

Dalam hal penggunaan sensor opto-coupler yang digunakan pada praktikum Modul 4 ini, sinyal yang dibangkitkan sudah dalam bentuk digital (0/LOW atau 1/HIGH). Sehingga tidak diperlukan pengondisi sinyal analog dan konverter analog ke digital dalam kasus ini.

- Periodic Timer: adalah komponen untuk memicu operasi sesuai selang waktu (interval) tertentu. Biasanya pengontrol-mikro selalu memiliki timer bawaan. Namun perlu diperhatikan bahwa timer hanya dapat memberikan waktu relatif sejak pengontrol-mikro di-reset. Jika akuisisi data memerlukan waktu absolut (sesuai tanggal dan jam di dunia nyata), maka diperlukan komponen real-time clock.
- Pencuplikan: bagian program yang akan dipicu oleh timer untuk mengukur (membaca) data.
- RingBuffer: data dari pencuplikan dimasukkan ke RingBuffer agar dapat diolah dengan mudah secara online.
- Pengolahan data digital: bertugas mengolah data yang tersimpan di RingBuffer agar diperoleh sinyal yang lebih baik. Dalam hal ini, akan digunakan *moving average*.
- Display: bagian program menampilkan hasil pengukuran, dalam hal ini Serial Plotter di komputer.

Selanjutnya bisa ditambahkan bagian kedua yang kita sebut Recorder, terdiri atas:

- FlashBuffer: menyimpan data ke EEPROM.

- EEPROM : komponen memori untuk menyimpan data secara tetap (tak hilang walau mikro-kontroller mati).

Bagian terakhir adalah Sender, terdiri atas:

- Sender: bagian program yang membaca data dari FlashBuffer dan mengirim / unggah data.
- Serial: perangkat keras untuk komunikasi data ke komputer.

4.5 TUGAS AWAL

Lakukan persiapan berikut ini:

- Unduh dan pasanglah pustaka TFRingBuffer dan TFlashBuffer.
- Unduh semua kode sumber modul 4.
- Carilah datasheet dan pin address driver motor L9110.
- Jelaskan cara kerja driver motor dengan rangkaian H-Bridge, bagaimana motor dapat bergerak searah putaran jarum jam dan berlawanan arah jarum jam menggunakan driver rangkaian H-Bridge ini?

Kemudian lakukan langkah-langkah sebagai berikut sebagai persiapan praktikum:

1. Cobalah buka program M4-01. Pelajari kode sumber, jelaskan bagaimana cara memesan buffer, menyimpan data, dan mengambil data.

   Coba jalankan, amati luarannya pada Serial monitor, akan tampak kolom sebagai berikut:

   S00 = sinyal pertama S(t)

   S01 = sinyal kedua S(t-1)

   S02 = sinyal kedua S(t-2)

   S03 = sinyal kedua S(t-3)

   S0A = Moving Average sinyal sesuai window tertentu (ada di program)
   S10 – S1A sama, untuk sinyal kedua.

S00	S01	S02	S03	S0A	S10	S11	S12	S13	S1A
00	00	00	00	00.00	10	00	00	00	05.00
01	00	00	00	00.50	10	10	00	00	10.00
02	01	00	00	01.50	10	10	10	00	10.00
03	02	01	00	02.50	10	10	10	10	10.00
04	03	02	01	03.50	10	10	10	10	10.00
05	04	03	02	04.50	00	10	10	10	05.00
06	05	04	03	05.50	00	00	10	10	00.00
07	06	05	04	06.50	00	00	00	10	00.00
08	07	06	05	07.50	00	00	00	00	00.00
09	08	07	06	08.50	00	00	00	00	00.00

   a. Amati apakah setiap periode, sinyal pada ring buffer akan bergeser?

   b. Amati apakah moving average memang sudah benar?

c. Coba edit kode sumber, perbaiki fungsi movAverage agar dapat menghitung untuk berbagai window (window > 1).
d. Coba program editan Anda. Nyatakan apakah Pustaka RingBuffer ini sudah benar, dan Anda sudah bisa membuat fungsi MovAverage dengan baik.

2. Cobalah buka program M4-02. Pelajari kode sumber, jelaskan bagaimana cara memesan buffer, menyimpan data, dan mengambil data.

   Jalankan program, kemudian luncurkan Serial monitor untuk lihat outputnya.

   a. Tekan Reset, amati tampilan awal tentang berbagai ukuran buffer untuk berbagai macam tipe data. Catat sebagai referensi anda.
   b. Masukkan 6 data atau lebih. Berapa data yang tersimpan?
   c. Matikan ESP32 (cabut USB-nya). Setelah beberapa saat hidupkan lagi.
   d. Tekan Reset.
   e. Amati apakah data yang tersimpan memang bisa dibaca kembali.
   f. Masukkan data kosong sampai buffer terhapus.
   g. Tekan Reset.
   h. Amati apakah buffer memang sudah terhapus.
   i. Nyatakan apakah EEPROM dan Pustaka FlashBuffer memang sudah siap bekerja dengan baik.

3. Pada tugas awal ini, kita akan mengatur nyala gelap-terang LED0 (LED yang built in pada EScope) yang terhubung pada pin GPIO 5 ESP32. Kemudian pada Arduino IDE, tuliskan kode sumber berikut ini.

```
/* Test PWM
 * Menyalakan LED menggunakan sinyal PWM
 */

// Memilih pin PWM
const int ledPin = 5; //pin GPIO 5 yang terhubung ke LED0 built in pada
EScope

// Pengaturan properti PWM
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

void setup(){
  // konfigurasi fungsi PWM LED
  ledcSetup(ledChannel, freq, resolution);

  // menempelkan channel ke GPIO yang akan dikontrol
  ledcAttachPin(ledPin, ledChannel);
}

void loop(){
  // Meningkatkan kecerahan LED
  for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
```

```
    // Mengubah kecerahan LED dengan PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }
  // Mengurangi kecerahan LED
  for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
    // Mengubah kecerahan LED dengan PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }
}
```

Unggah kode di atas ke ESP32, dan amati yang terjadi pada LED, kemudian laporkan pada tugas awal.

4.6 PRAKTIKUM

4.6.1 MOVING AVERAGE

Pada percobaan ini, kita akan mengamati efektifitas moving average untuk pengolahan data, khususnya melakukan penapisan sinyal. Untuk itu telah dibuat program M4-03 yang :

- membangkitkan dua sinyal, yaitu kotak dan sinus, dengan frekuensi tertentu. Sinyal sinus akan diberi "noise" berupa dua sinyal sinus lain yang frekuensinya lebih tinggi.
- Kedua sinyal akan dilewatkan ke fungsi movAverage, dengan window yang berbeda.
- Semua sinyal diplot sehingga dapat diamati hasilnya.

Ingin ditemukan nilai window yang tepat agar sinyal sinus asli bisa didapat kembali, namun sambil mengamati dampak lain yang mungkin terjadi.

Gambar 4.18. Contoh capture layar moving average

Prosedur percobaan adalah sebagai berikut:

1. Ambil kode sumber M4-03
2. Perbaiki fungsi movAverage sesuai tugas awal anda.
3. Jalankan program, capture screen dan bandingkan sinyal S1 (sudah kena noise) dan S2 (sinyal sinus tanpa noise). Sementara itu A11, A12, dan A13 adalah sinyal setelah diolah dengan moving average dengan window tertentu. Amati, bagaimana noise menghilang pada A11-A13.
4. Coba edit kode M4-03, ubah WINDOW_1 … WINDOW_3 (dan N_ROW jika perlu). Jalankan Kembali program, cobalah cari hingga bentuk A13 mendekati S3 (noise hilang).
5. Simpulkan berapa harga N_WINDOW yang diperlukan untuk menghilangkan noise ?
6. Coba kaitkan harga WINDOW dengan frekuensi kedua sinus noise.
7. Sebagai dampak, apa yang terjadi dengan sinyal seiring membesarnya WINDOW ?
8. Amati sinyal kotak, jelaskan mengapa pada A01 .. A03 terjadi ramp yang berbeda ? Apa hubungannya dengan WINDOW ?

4.6.2 PENGUJIAN OPTO-COUPLER

Pada percobaan ini, akan diuji apakah opto-coupler bisa mendeteksi baling-baling motor DC. Untuk itu akan baling-baling digerakkan manual dengan tangan, kemudian sinyal pulsa opto-coupler (ON/OFF) ditampilkan pada grafik di Serial Plotter yang menunjukkan perbedaan saat opto-coupler terhalang dan tidak terhalang oleh bilah motor DC. Prosedur percobaan:

1. Ambil kode sumber M4-04-OptoCoupler, jalankan.
2. Luncurkan Serial Plotter
3. Memakai tangan, perlahan putar baling-baling agar TIDAK menutupi opto-coupler. Perhatikan pada Serial Plotter, PULSA seharusnya LOW.
4. Putar baling-baling agar menutupi opto-coupler. Sinyal PULSA seharusnya HIGH.
5. Putar baling-baling beberapa kali. Perhatikan pada legend Serial plotter, Counter seharusnya naik setiap kali bilah menutupi opto-coupler.

Jika pada prosedur tersebut pulsa tidak sesuai, cobalah mencari lingkungan yang cahayanya tidak terlalu terang. Kalau masih belum berhasil juga, opto-coupler perlu dikalibrasi sensitivitasnya dengan memutar trimpot-nya. Lanjutkan praktikum hanya jika opto-coupler sudah bekerja dengan baik.

Selenjutnya pada percobaan 4.6.3 sampai 4.6.7 akan dicoba mengatur kecepatan motor DC dengan mengatur PWM, lalu kecepatan motor diukur oleh opto-coupler. Dalam hal ini ada 4 percobaan yang akan dilakukan, yaitu :

a) Mengamati pengaruh duty-cycle PWM terhadap kecepatan motor DC
b) Mengamati pengaruh frekuensi PWM terhadap kecepatan motor DC
c) Meningkatkan kecepatan ukur dan ketelitian opto-coupler dengan moving average
d) Mengontrol kecepatan motor DC dengan kontrol PID

4.6.3 PENGATURAN KECEPATAN MOTOR DC DENGAN DUTY CYCLE PWM

Untuk percobaan a), kita akan mengambil data RPM terhadap PWM. Untuk itu siapkan tabelnya untuk mencatat, lalu lakukan sebagai berikut:

1. Ambil kode sumber M4-05-DC_Motor_RPM. Perbaiki fungsi optoGetRPM() agar diperoleh pengukuran RPM yang benar.
2. Jalankan program tersebut, lalu luncurkan Serial Monitor.

3. Kita akan masukkan beberapa harga PWM. Setiap kali, amati luaran di Serial Monitor sampai relative tunak, catat harga PWM dan RPM hasil pengukuran.
4. Mulailah dengan memasukkan harga PWM dari kecil, misal 10, amati apakah motor sudah mulai berputar. Coba naikkan harga PWM sampai motor mulai berputar. Catat itu sebagai PWM_A.
5. Ulangi percobaan dengan menaikkan harga PWM dari PWM_A ke 255, dengan selang tetap sekitar ((255-PWM_A)/8) sehingga didapat 8 data.
6. Cobalah sebaliknya, kini turunkan PWM dari 255 menuju ke PWM_A, sampai motor berhenti.
7. Setelah itu kita akan coba arah putaran sebaliknya.
8. Masukkan harga PWM dari NEGATIF kecil, misal -10, turunkan sampai sampai motor mulai berputar. Catat itu sebagai -PWM_B.
9. Ulangi pengamatan dengan menurunkan harga PWM dari -PWM_B ke -255, dengan selang tetap sekitar ((255-PWM_A)/8) sehingga didapat 8 data.
10. Cobalah sebaliknya, kini naikkan PWM dari -255 menuju ke PWM_B sampai motor berhenti.

Dari catatan percobaan ini, buatlah grafik RPM terhadap PWM. Lakukan analisis untuk menentukan:

- Berapa nilai PWM deadtime (PWM terkecil yang membuat motor bekerja)
- Berapa nilai maksimum RPM ?
- Apakah hubungan input/output RPM dan PWM cukup linier ?
- Apakah ada bedanya hubungan input/output tersebut saat PWM naik / turun ?
- Kaji mengapa kelakuan motor DC seperti itu.

4.6.4 PENGARUH FREKUENSI PWM TERHADAP KECEPATAN MOTOR DC

Kini kita akan mengamati pengaruh frekuensi PWM terhadap kecepatan motor DC, sementara duty cycle PWM tetap sebesar 50% (128). Untuk percobaan ini siapkan tabelnya untuk mencatat FREQ dan RPM. Lakukan sebagai berikut:

1. Ambil kode sumber M4-06-DC_Motor _FREQ. Perbaiki fungsi optoGetRPM() agar diperoleh pengukuran RPM yang benar.
2. Jalankan program tersebut, lalu luncurkan Serial Monitor.
3. Kita akan masukkan beberapa harga FREQ. Setiap kali, amati luaran di Serial Monitor sampai relative tunak, catat harga PWM dan RPM hasil pengukuran.
4. Mulailah dengan memasukkan harga FREQ kecil. Amati putaran motor.
5. Naikkan harga FREQ secara berlipa, misalnya 1, 100, 200, 400, 800, 1000, 2000. Setiap kali amati putaran motor. Cobalah sampai motor terlihat melambat.

Dari catatan percobaan ini, buatlah grafik FREQ terhadap PWM. Lakukan analisis untuk menentukan:

- Apakah ada perbedaan antara FREQ kecil dengan besar ?
- Berapa rentang FREQ dimana putaran motor masih baik.
- Kaji mengapa kelakuan motor DC seperti itu.

4.6.5 PENGUKURAN KECEPATAN MOTOR DC DENGAN MOVING AVERAGE

Pada dua percobaan sebelumnya, RPM dihitung dengan selang waktu pengukuran yang cukup besar (10 detik). Hal ini membuat ketelitian pengukuran menjadi tinggi, namun pengukuran menjadi lambat (tidak responsive) sehingga hanya dapat digunakan untuk pengukuran kecepatan motor sudah tunak. Untuk meningkatkan

kecepatan pengukuran, tanpa mengurangi ketelitian, maka bisa digunakan metode moving average (MA). Lakukan sebagai berikut:

1. Ambil kode sumber M4-07-DC_Motor_Opto_MA. Coba pahami proses pengukuran dengan MA tersebut. Perbaiki fungsi optoGetRPM() sedikit agar diperoleh pengukuran RPM yang benar.
2. Jalankan program tersebut, lalu luncurkan Serial Plotter.
3. Coba masukkan suatu harga RPM yang besar (misal 200).
4. Amati di serial plotter, bagaimana pengukuran kecepatan bisa mengikuti respon transien motor. Capture layar.
5. Tunggu sampai tunak, amati hasil pengukuran RPM dan catat.
6. Coba turunkan lagi RPM yang kecil (misal PWM_A dari percobaan sebelumnya).
7. Amati respon transien motor, capture screen.
8. Tunggu sampai tunak, amati hasil pengukuran RPM.

Dari percobaan tersebut cobalah:

- Kaji apakah MA memang cukup bisa mengukur respon transien. Jika belum bisa, apa yang perlu diperbaiki?
- Kaji apakah hasil pengukuran MA memang cukup teliti, setara dengan pengukuran sebelumnya yang intervalnya besar?
- Jelaskan cara kerja pengukuran kecepatan motor dengan opto dan MA ini. Apa kira-kira pengaruh N_WINDOW dan periode ukur (ts_measure) pada pengukuran ini?

4.6.6 PENGONTROLAN MOTOR DC DENGAN MOVING AVERAGE DAN KONTROL PID

Pada percobaan ini, kita akan mengontrol kecepatan motor DC yang nantinya diperintah melalui Serial Plotter.

1. Buka kode M4-08-DC_Motor_Opto_MA_PID dan unggah ke ESP32 Anda.
2. Luncurkan Serial Plotter.
3. Masukkan kecepatan yang diinginkan dalam RPM (disebut sebagai *set point*/SP) pada Serial Plotter, kemudian Enter.
4. Amati di Serial Plotter, bagaimana pengontrolan kecepatan terjadi. Capture layar.
5. Tunggu sampai tunak, amati hasil pengontrolan motor dan catat.
6. Coba turunkan lagi RPM yang kecil.
7. Tunggu sampai tunak kembali.
8. Amati hasil pengontrolan motor, capture screen.

Berikut ini adalah cara kerja program M4-08-DC_Motor_Opto_MA_PID:

- Program akan berusaha memutar motor DC sesuai dengan permintaan *set point*.
- Antara *set point* dengan kecepatan RPM motor DC terdapat perbedaan. Perbedaan antara kecepatan dengan set point menjadi nilai *error*.
- Program akan secara otomatis mengubah nilai PWM (menaikkan atau menurunkan PWM) sesuai dengan nilai error yang sudah dihitung dan memasukkannya ke dalam rumus kontrol PID.

Coba ubah kinerja pengukuran dengan mengganti `ts_measure` dan `ts_display`, juga nilai `pid_kp`, `pid_ki`, dan `pid_kd`.

4.7 TUGAS & LAPORAN

Untuk membuat laporan jawab semua pertanyaan yang ada di atas. Untuk tugasnya buat flowchart untuk setiap kode.

4.8 PUSTAKA

- Pulse Width Modulation (PWM) : https://www.electronicshub.org/esp32-pwm-tutorial/ ; https://randomnerdtutorials.com/esp32-pwm-arduino-ide/
- Interrupt: https://www.theengineeringprojects.com/2021/12/esp32-interrupts.html
- ADC : https://docs.espressif.com/projects/esp-idf/en/release-v4.2/esp32s2/apireference/peripherals/adc.html
- ADC: https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/
- Arduino timing : https://www.programmingelectronics.com/millis-arduino/
- OLED: https://microcontrollerslab.com/oled-display-arduino-tutorial/
- EEPROM : https://pijaeducation.com/eeprom-in-arduino-and-esp
- Motor DC: https://randomnerdtutorials.com/esp32-dc-motor-l298n-motor-driver-control-speed-direction/
- Sensor Opto-Coupler: https://www.instructables.com/How-to-Use-TCRT5000-IR-Sensor-Module-With-Arduino-/ ; https://diyi0t.com/tcrt5000-line-tracking-module-arduino-esp8266-esp32/