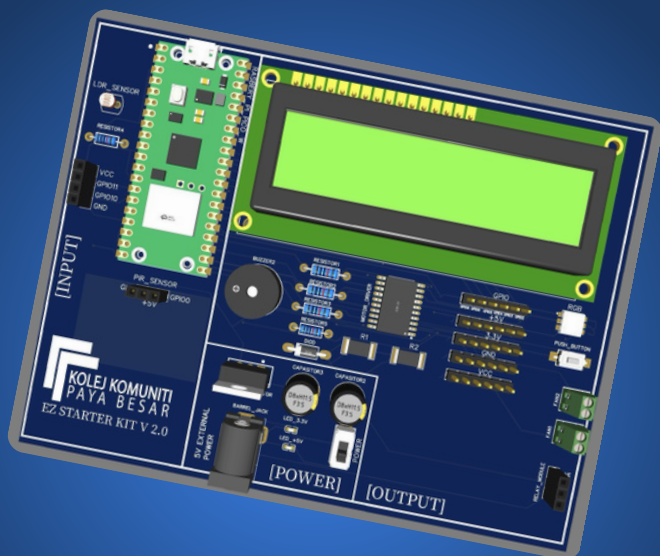


MANUAL

PENGGUNAAN

EZ STARTER KIT VERSION 2.0



NURUL NAJIAH BINTI MOHAMMAD
HAFIZAH BINTI ISMAIL
DZAZURIATY BINTI MUHAMAD



HAK CIPTA TERPELIHARA

Edisi Pertama 2026

Hak Cipta Kolej Komuniti Paya Besar.

Hak Cipta Terpelihara. Sebarang pengeluaran semula mana-mana bahagian daripada penerbitan ini dalam apa jua bentuk atau melalui apa jua kaedah, sama ada secara elektronik, fotokopi, rakaman atau mekanikal, tanpa kebenaran bertulis terlebih dahulu daripada Unit Pembelajaran Digital, Kolej Komuniti Paya Besar adalah dilarang sama sekali.

Penulis:

- Nurul Najihah binti Mohammad
- Hafizah binti Ismail
- Dzazuriaty binti Muhamad

eISBN: 978-629-94331-8-7

Diterbitkan oleh:
Kolej Komuniti Paya Besar
(Unit Pembelajaran Digital)
Kementerian Pendidikan Tinggi Malaysia
26300 Gambang, Pahang

Laman Web: <https://kkpayabesar.mypolycc.edu.my>



Data Pengkatalogan-dalam-Penerbitan

Perpustakaan Negara Malaysia

Rekod katalog untuk buku ini boleh didapati dari
Perpustakaan Negara Malaysia

eISBN 978-629-94331-8-7



PEMBANGUN



**NURUL NAJIHAH BT
MOHAMMAD**

Pensyarah
Program Sijil Teknologi Maklumat
Kolej Komuniti Paya Besar



**HAFIZAH BINTI
ISMAIL**

Pensyarah
Program Sijil Teknologi Maklumat
Kolej Komuniti Paya Besar
Cawangan Maran



**DZAZURIATY BINTI
MUHAMAD**

Pensyarah
Program Sijil Teknologi Maklumat
Kolej Komuniti Paya Besar



SINOPSIS

EZ Starter Kit Version 2.0 merupakan sebuah kit pembelajaran yang direka khas untuk memperkenalkan konsep asas Internet of Things (IoT) secara praktikal dan interaktif. Kit ini dibangunkan sebagai platform pembelajaran yang mesra pengguna bagi membantu pelajar, pensyarah serta pemula memahami bagaimana peranti fizikal boleh berhubung, berkomunikasi dan dikawal melalui rangkaian internet.

Kit ini menggunakan mikropengawal seperti Raspberry Pi Pico W yang mempunyai keupayaan sambungan WiFi terbina, membolehkan integrasi terus dengan pelbagai aplikasi IoT. Ia dilengkapi dengan pelbagai komponen seperti sensor ultrasonik, sensor gerakan (PIR), paparan LCD, modul relay, motor driver serta komponen sokongan lain yang membolehkan pelbagai eksperimen dan projek dijalankan.

Reka bentuk EZ Starter Kit Version 2.0 adalah berasaskan pembelajaran berasaskan projek (project-based learning), di mana pengguna bukan sahaja mempelajari teori, tetapi turut membina sistem sebenar seperti pemantauan persekitaran, automasi peranti dan sistem kawalan pintar. Selain itu, kit ini turut menyokong integrasi dengan platform digital untuk tujuan pemantauan data secara masa nyata.

Secara keseluruhannya, EZ Starter Kit Version 2.0 bertujuan untuk menyediakan asas kukuh dalam bidang IoT serta meningkatkan kemahiran teknikal, kreativiti dan pemikiran kritis dalam kalangan pengguna sejajar dengan keperluan teknologi masa kini.



OBJEKTIF

1

Mengenali komponen asas IoT

2

Mempelajari sambungan perkakasan dan perisian

3

Membina projek IoT mudah



ISI KANDUNGAN



Pengenalan kepada komponen IOT

M/S 6 - 26



Spesifikasi Komputer

M/S 27 - 28



Penyediaan Perisian

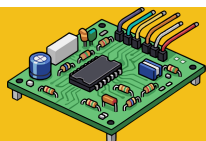
M/S 29 - 32



Modul Aktiviti

M/S 36 - 74

Pengenalan kepada Komponen IOT



KOMPONEN IOT

1

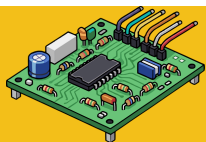
Raspberry Pi Pico W

- Raspberry Pi Pico W ialah sebuah papan mikropengawal bersaiz kecil yang dibangunkan oleh Raspberry Pi Foundation dan direka khusus untuk projek elektronik serta Internet of Things (IoT).
- Versi “W” merujuk kepada keupayaan WiFi terbina dalam, yang membolehkan papan ini berhubung terus ke rangkaian tanpa memerlukan modul tambahan.

- Menggunakan cip RP2040 (dual-core ARM Cortex-M0+)
- Kelajuan sehingga 133 MHz
- Memori SRAM 264KB
- Storan flash 2MB
- Sambungan WiFi 2.4GHz (802.11n)
- 26 pin GPIO (boleh digunakan untuk input/output)
- Sokongan komunikasi:
 - I2C (Inter-Integrated Circuit)
 - SPI (Serial Peripheral Interface)
 - UART (Universal Asynchronous Receiver-Transmitter)
- Voltan operasi: 3.3V



Ciri-ciri Utama



KOMPONEN IOT

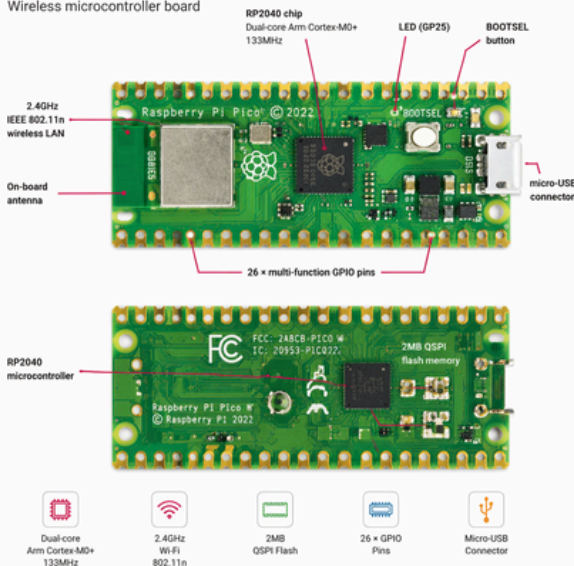


Kelebihan

- Boleh hantar dan terima data melalui internet
- Sesuai untuk sistem pemantauan (monitoring system)
- Menyokong kawalan peranti secara jarak jauh
- Integrasi dengan platform seperti Blynk, MQTT, dan ThingSpeak

Raspberry Pi Pico W

Wireless microcontroller board

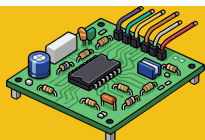


Raspberry Pi

KEY FEATURES

- RP2040 microcontroller chip designed by Raspberry Pi in the UK
- Dual-core Arm Cortex-M0+ processor, flexible clock running up to 133MHz
- 264KB on-chip SRAM, 2MB on-board QSPI flash
- 2.4GHz wireless LAN (802.11n)
- On-board PCB antenna
- 26 x multi-function GPIO pins
- USB 1.1 micro-B
- Drag-and-drop programming using mass storage over USB
- Low power sleep and dormant modes

Imej janaan AI



KOMPONEN IOT

2

Sensor : Ultrasonic (HC-SR04)

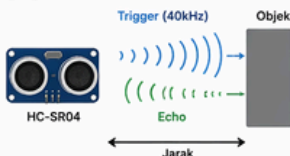
HC-SR04 ialah sensor yang digunakan untuk mengukur jarak sesuatu objek menggunakan gelombang ultrasonik.

Ultrasonic (HC-SR04)



CARA KERJA

Sensor menghantar gelombang ultrasonik (40kHz) dan menunggu pantulan (echo). Jarak dikira berdasarkan masa yang diambil oleh gelombang untuk pergi dan kembali.



SPESIFIKASI

- Voltan Operasi: 5V DC
- Arus Operasi: 15mA
- Frekuensi Ultrasonik: 40kHz
- Jarak Pengesanan: 2cm – 400cm
- Sudut Pengesanan: < 15°
- Isyarat Trigger: 10µs TTL pulse
- Isyarat Echo: Pulse TTL
- Dimensi: 45mm x 20mm x 15mm

PINOUT

Pin	Nama	Keterangan
1	VCC	5V DC
2	TRIG	Input trigger (10µs pulse)
3	ECHO	Output echo pulse
4	GND	Ground (0V)

APLIKASI



Pengukur Jarak



Sistem Parkir

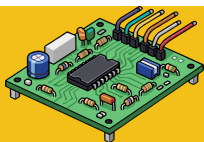


Elak Halangan Robot



Sistem Tong Sampah Pintar

Imej janaan AI



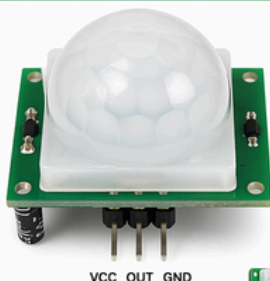
KOMPONEN IOT

3

Sensor : PIR Motion Sensor

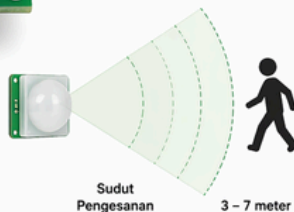
- PIR (Passive Infrared) digunakan untuk mengesan pergerakan manusia atau haiwan berdasarkan perubahan haba (infrared).

PIR Motion Sensor



CARA KERJA

Sensor mengesan perubahan haba inframerah pada badan manusia atau haiwan yang bergerak dalam kawasan pengesanan.



SPESIFIKASI

- Voltan Operasi: 5V DC
- Arus Operasi: 50 – 65µA
- Jarak Pengesanan: 3 – 7 meter
- Sudut Pengesanan: < 120°
- Output: Digital HIGH (3.3V) / LOW (0V)
- Masa Tunda: 5 – 200 saat (laras)
- Dimensi: 32mm x 24mm x 18mm

PINOUT

Pin	Nama	Keterangan
1	VCC	5V DC
2	OUT	Output Digital
3	GND	Ground (0V)

APLIKASI



Lampu Automatik



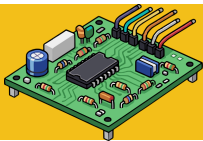
Sistem Penggera



Kipas / Peranti Automatik



Pengesan Kehadiran



KOMPONEN IOT

4

Output : LCD I2C 16x2

- LCD 16x2 ialah paparan yang boleh memaparkan 16 aksara $\times$ 2 baris, manakala modul I2C membolehkan sambungan lebih mudah menggunakan hanya 2 pin komunikasi.



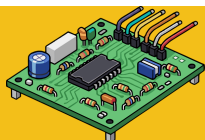
Ciri-ciri Utama

- Paparan: 16 aksara $\times$ 2 baris
- Antara muka: I2C (SDA: (Serial Data Line) & SCL: SCL (Serial Clock Line))
- Voltan operasi: 5V
- Jimat pin (hanya 4 sambungan):
 - VCC
 - GND
 - SDA
 - SCL



Fungsi

- Memaparkan data sensor (jarak, status, suhu dll)
- Paparan mesej sistem (contoh: "System ON")
- Output visual untuk pengguna



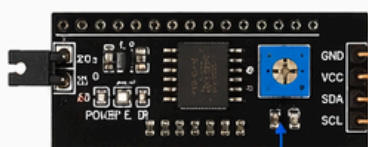
KOMPONEN IOT

4

Output : LCD I2C
16x2

LCD I2C 16x2

Paparan 16 aksara x 2 baris dengan modul I2C



Laraskan kontras

SAMBUNGAN (I2C)

LCD I2C	MIKROPENGAWAL
VCC	5V
GND	GND
SDA	SDA (A4)
SCL	SCL (A5)

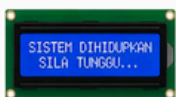
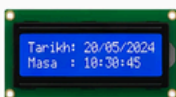
SPESIFIKASI

- Voltan Operasi: 5V DC
- Antara muka: I2C (Alamat default: 0x27 / 0x3F)
- Paparan: 16 aksara x 2 baris
- Warna Backlight: Biru / Hijau (bergantung model)
- Kawalan kecerahan melalui perisian (backpack)
- Dimensi Modul: 80mm x 36mm x 13mm

CIRI-CIRI

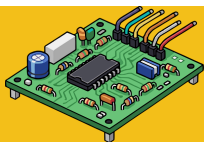
-  Hanya 2 pin I2C diperlukan (SDA, SCL)
-  Backlight boleh dikawal melalui perisian
-  Mudah digunakan dengan Arduino, Raspberry Pi, dll.

CONTOH PAPARAN



* Paparan mungkin berbeza mengikut model dan pengeluar.

Imej janaan AI



KOMPONEN IOT

5

Output : Mini DC Motor Fan

- Mini DC Motor Fan ialah bagi menunjukkan konsep kawalan motor dalam aplikasi Internet of Things (IoT).



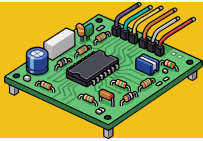
Ciri-ciri Utama

- Jenis: Mini DC Motor Fan / Mini Propeller Fan
- Bilah kipas: 3 bilah
- Motor: Motor DC kecil
- Wayar: Merah (+) dan hitam (-)
- Bekalan: Nampaknya direka untuk bekalan DC rendah, tetapi voltan tepat perlu dirujuk pada spesifikasi motor sebenar.
- Kawalan: Dikawal melalui litar motor driver pada EZ Starter Kit
- Fungsi: Menghasilkan aliran udara sebagai output dalam projek automasi IoT



Fungsi

- Fan ini menggunakan motor DC kecil yang dikawal oleh Raspberry Pi Pico W melalui litar motor driver L293D.



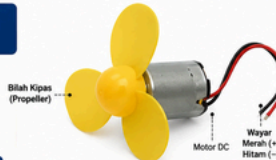
KOMPONEN IOT

5

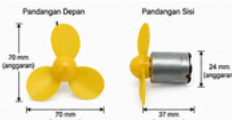
Output : Mini DC Motor Fan

MINI DC MOTOR FAN (Kipas Mini DC Jenis Propeller)

Mini DC Motor Fan merupakan perangkat output yang menghasilkan aliran udara. Ia menggunakan motor DC kecil yang dipasang bersama bilah kipas (propeller). Fan ini dikawal oleh mikropengawal melalui litar driver motor.



DIMENSI (Anggaran)



* Saiz mungkin berbeza mengikut model sebenar.

SPEKIFIKASI TEKNIKAL

Voltag Operasi	DC 3V – 6V
Arus Operasi	± 100 mA – 300 mA
Jenis Motor	Brushed DC Motor
Bilangan Bilah	3 bilah
Jenis Sambungan	2-wayer (Merah +, Hitam -)
Arah Putaran	Sehala (berubah mengikut polariti)
Bahan Bilah	Plastik
Aplikasi	Projek elektronik, IoT, sistem pengendalian mini, demonstrasi

CIRI-CIRI

- Jenis : Mini DC Motor Fan (Propeller)
- Bilah : 3 bilah
- Motor : DC 3V – 6V (bergantung model)
- Arus operasi : ± 100 mA – 300 mA (anggaran)
- Arah putaran : Sehala (bergantung polariti)
- Wayer : 2 wayer (Merah = +, Hitam = -)
- Fungsi : Menghasilkan aliran udara sebagai output sistem

APLIKASI



FUNGSI DALAM SISTEM

Fan dikawal oleh HODUP atau MATI berdasarkan input daripada sensor atau arahan daripada sistem IoT.



SAMBUNGAN WAYER

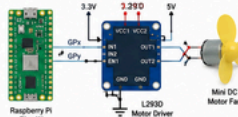
Wayer	Sambungan
Merah (+)	Ka pin output motor driver (+)
Hitam (-)	Ka pin output motor driver (-)

Nota:
Tukar polariti wayer (jika arah putaran kipas perlu dibalikkan (jika motor menyokong)).



CARA KERJA

1. Mikropengawal memberikan isyarat kepada motor driver (L293D).
2. Motor driver membekalkan voltan kepada motor DC.
3. Motor DC berputar dan memutarakan bilah kipas (propeller).
4. Fan menghasilkan aliran udara.



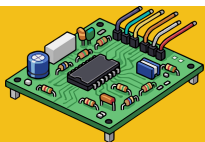
LANGKAH KESELAMATAN

- ⚠ Pastikan polariti sambungan beluli (Merah ke + dan Hitam ke -).
- ⚠ Jangan beri voltan melebihi spesifikasi motor (maks. 6V).
- ⚠ Jangan sentuh bilah kipas ketika ia berputar.
- ⚠ Pastikan kipas diletakkan pada permukaan yang stabil.

CONTOH PENGGUNAAN DALAM EZ STARTER KIT V2.0



Imej janaan AI



KOMPONEN IOT

6

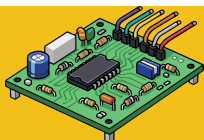
Komponen lain : BUZZER

- Buzzer ialah komponen elektronik yang digunakan untuk menghasilkan bunyi atau isyarat audio apabila menerima isyarat elektrik. Dalam EZ Starter Kit Version 2.0, buzzer berfungsi sebagai salah satu peranti output yang boleh dikawal oleh Raspberry Pi Pico W.



Fungsi

- Buzzer boleh digunakan untuk memberikan pemberitahuan kepada pengguna apabila sesuatu keadaan berlaku, contohnya:
- 🚨 Memberi amaran apabila pergerakan dikesan oleh PIR.
- 📏 Memberi bunyi apabila objek berada pada jarak tertentu berdasarkan sensor ultrasonik.
- 🔔 Memberikan notifikasi apabila sesuatu proses selesai.
- ⚠ Memberi amaran apabila keadaan sistem tidak normal.
- 🤖 Digunakan sebagai output dalam projek automasi IoT.



KOMPONEN IOT

6

Komponen lain : BUZZER

BUZZER

EZ Starter Kit Version 2.0



ACTIVE BUZZER
(Tidak perlu isyarat frekuensi)



PASSIVE BUZZER
(Perlu isyarat frekuensi / PWM)

APA ITU BUZZER?

Buzzer ialah komponen elektronik yang menghasilkan bunyi apabila menerima isyarat elektrik. Ia digunakan sebagai amaran, notifikasi atau indikator dalam projek elektronik dan IoT.

FUNGSI BUZZER

- ✓ Memberi amaran apabila persediaan dihentikan.
- ✓ Memberi bunyi apabila jarak objek tertentu.
- ✓ Memberi notifikasi apabila proses selesai.
- ✓ Amaran apabila keadaan sistem tidak normal.
- ✓ Output audio dalam sistem automasi IoT.

SPESIFIKASI TEKNIKAL

Jenis	Active / Passive
Voltag Operasi	3.3V ~ 5V DC
Anus Operasi	± 10mA ~ 30mA (anggaran)
Frekuensi Bunyi	2.0 ~ 4.0 kHz (Active) 1 ~ 5 kHz (Passive)
Aras Bunyi	± 85 dB pada 10cm (anggaran)
Sambungan	2 Pin
Kawalan	GPIO / PWM
Peranan	Output / Indikator Audio
Penggunaan	Amaran, notifikasi, status sistem

JENIS BUZZER

1. ACTIVE BUZZER

- Bunyi dihasilkan dengan tanpa beri isyarat HIGH/LOW.
- Tidak perlu isyarat frekuensi.
- Mudah digunakan.



2. PASSIVE BUZZER

- Bunyi dihasilkan dengan isyarat berfrekuensi (PWM / frekuensi).
- Boleh dihasilkan nada berbeza.
- Lebih fleksibel.



SAMBUNGAN PIN

ACTIVE BUZZER



PASSIVE BUZZER

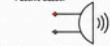


SIMBOL LITAR

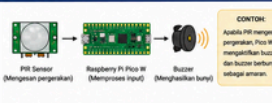
Active Buzzer



Passive Buzzer



CARA KERJA



CONTOH

Apabila PIR mengesan pergerakan, Pico W mengaktifkan buzzer dan buzzer berbunyi sebagai amaran.

CONTOH SAMBUNGAN DENGAN RASPBERRY PI PICO W



APLIKASI



PERHATIAN

- Pastikan potensi sambungan betul.
- Jangan beri voltan melebihi had operasi buzzer.

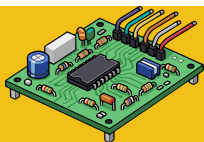
- Gunakan perintang jika disambungkan terus ke GPIO (berdaya 3.3V).
- Elakkan sambungan terus tanpa perintang pada buzzer jenis Passive.



TIP

Untuk Passive Buzzer: Gantikan isyarat PWM pada frekuensi 20kHz ~ 50kHz untuk menghasilkan bunyi.

Imej janaan AI



KOMPONEN IOT

7

Komponen lain : LDR Sensor

- LDR (Light Dependent Resistor) ialah komponen elektronik yang digunakan untuk mengesan tahap keamatan cahaya. Nilai rintangan LDR berubah mengikut jumlah cahaya yang diterima.



Fungsi

- Mengesan keadaan cerah atau gelap
- Mengawal lampu secara automatik
- Mengaktifkan buzzer berdasarkan tahap cahaya
- Mengawal peranti berdasarkan keadaan persekitaran
- Menghantar bacaan tahap cahaya ke platform IoT



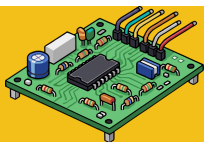
Cara Kerja

- ☀️ Cahaya terang → rintangan LDR menurun
- 🌙 Cahaya semakin gelap → rintangan LDR meningkat



Pemasangan

- LDR disambungkan dalam litar pembahagi voltan (voltage divider) bersama resistor.



KOMPONEN IOT

7

Komponen lain : LDR Sensor

LDR SENSOR (Light Dependent Resistor) EZ Starter Kit Version 2.0

LDR ialah komponen yang peka terhadap cahaya. Nilai rintangan LDR berubah mengikut keamatan cahaya yang diterima.

CARA LDR BERFUNGSI

Cahaya Terang

Rintangan menurun (Ohm rendah)

Cahaya Gelap

Rintangan meningkat (Ohm tinggi)

Lebih banyak cahaya → Rintangan lebih rendah
Lebih sedikit cahaya → Rintangan lebih tinggi

SPEKIFIKASI LDR SENSOR

Jenis	LDR (Light Dependent Resistor)
Jumlah Rintangan	~ 1 kΩ (terang) hingga ~ 1 MΩ (gelap)
Voltan Operasi	3.3V - 5V DC
Arus Operasi	< 1 mA
Output	Analog (perubahan voltan)
Sambungan	2 Pin (tidak polariti)
Peneraan	Input Sensor
Aplikasi	Pengesan cahaya, lampu automatik, sistem IoT, pemantauan persekitaran

BENTUK FIZIKAL LDR

LITAR PEMBAHAGI VOLTAN (VOLTAGE DIVIDER)

3.3V / 5V

LDR

Resistor (10kΩ)

GND

Vout (ke ADC Pico W)

PRINSIP: LDR dan resistor membentuk litar pembahagi voltan. Voltan output (Vout) berubah mengikut cahaya yang diterima dan dibina oleh ADC pada Raspberry Pi Pico W.

TINGKAH LAKU OUTPUT

Kedudukan Cahaya	Rintangan LDR	Voltan Output (Vout)
Terang	Rendah (~ 1 kΩ - 10 kΩ)	Tinggi (hampir kepada 3.3V / 5V)
Gelap	Tinggi (~ 100 kΩ - 1 MΩ)	Rendah (hampir kepada 0V)

Voltan yang berubah ini akan diukur kepada nilai digital oleh ADC Pico W (0 - 4095 bagi 12-bit ADC).

SAMBUNGAN PIN

LDR	SAMBUNGAN
Pin 1	Ke 3.3V / 5V
Pin 2	Ke Resistor (10kΩ) kemudian ke GND (dan tidak langsung ke ADC Pico W)

Nota:

- LDR tidak polariti. Boleh pasang mana-mana pin.
- Pastikan sambungan kukuh pada breadboard / PCB.

CONTOH PENGGUNAAN DENGAN RASPBERRY PI PICO W

Contoh Aplikasi: LDR digunakan untuk mengesan tempoh senja automatik.

Jika cahaya terang (nilai ADC tinggi) → Lampu OFF

Jika cahaya gelap (nilai ADC rendah) → Lampu ON

CARA MEMBACA NILAI LDR

- Pico W membaca nilai analog di pin ADC (0 - 4095).
- Nilai lebih tinggi = lebih banyak cahaya.
- Nilai lebih rendah = lebih gelap.

Nilai ADC (12-bit)

0 → Gelap → 4095 → Terang

Contoh Bacaan:

- Nilai 100 - 1000 → Gelap
- Nilai 1000 - 3000 → Sedehang
- Nilai 3000 - 4095 → Terang

APLIKASI

Lampu Automatik

Sistem Rumah Peka (IoT)

Pemantauan Persekitaran

Sistem Aliran

SAIZ FIZIKAL (Anggaran)

5 mm

4 mm

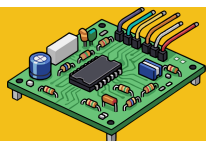
25 mm

Stemeter	5 mm
Tebali	4 mm
Panjang Pin	± 25 mm

PERHATIAN

- Jangan terkena cahaya sangat kuat dalam masa lama.
- Elakkan tayu pin ke atas sambungan.
- Gantikan nilai resistor yang sesuai (contoh: 10kΩ).

Imej janaan AI



KOMPONEN IOT

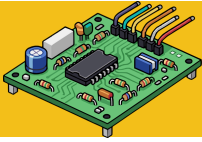
8

Komponen lain : RGB Digital

- Berbeza dengan LED RGB analog biasa yang hanya boleh memaparkan satu warna pada keseluruhan jalur pada satu-satu masa, LED RGB digital membolehkan anda mengawal setiap satu unit LED secara berasingan

Spesifikasi

Perkara	Spesifikasi Umum	Penerangan
Pakej LED	5050 SMD	Berukuran 5.0mm x 5.0mm, sangat padat.
Voltan Kendalian	5V DC	Memerlukan bekalan kuasa 5 Volt yang stabil.
Antaramuka	Protokol Tunggal (Single-wire)	Hanya memerlukan 1 pin data (DIN) untuk kawalan.
** Bilangan Pin**	3 Pin Utama	+5V (VCC), Ground (GND), dan Data Input (DIN).
Kedalaman Warna	24-bit True Color	8-bit setiap warna (Merah, Hijau, Biru).
Resolusi Kecerahan	256 Tahap per Warna	Boleh dilaras dari 0 (mati) hingga 255 (paling terang).
Jumlah Warna	16,777,216 Warna	Hasil kombinasi 256 x 256 x 256 warna.
Kelajuan Data	800 Kbps	Kelajuan penghantaran isyarat data serikal.
Frekuensi Imbasan	≥ 400 Hz/s	Memastikan pencahayaan bebas kelipan.
Suhu Operasi	-25°C hingga +80°C	Julat suhu kendalian yang luas.



KOMPONEN IOT

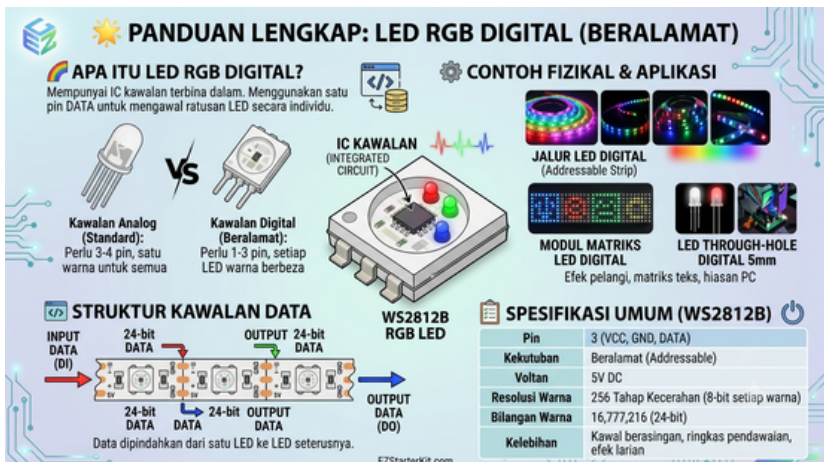
8

Komponen lain : RGB Digital

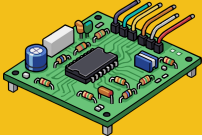


Kelebihan

- Kecerahan dan Konsistensi Warna yang Tinggi
 - Cip IC yang terbina dalam memastikan bekalan arus ke setiap chip warna (R, G, B) dalam LED adalah stabil dan konsisten. Ini menghasilkan warna yang sekata di seluruh jalur tanpa tompok kecerahan yang berbeza, asalkan bekalan kuasa 5V adalah mencukupi.



Imej janaan AI



KOMPONEN IOT

9

Komponen lain : Bateri Boleh Cas

- Bateri boleh cas semula (juga dikenali sebagai bateri sekunder) ialah sejenis bateri elektrik yang boleh dicas semula, digunakan semula, dan dinyahcas berkali-kali. Ini berbeza dengan bateri primer (pakai buang) yang hanya boleh digunakan sekali sehingga tenaga kimianya habis, kemudian perlu dibuang.

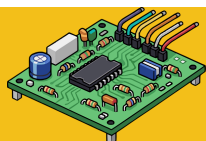
Cara Berfungsi:



Prinsip asas bateri adalah menyimpan tenaga elektrik dalam bentuk tenaga kimia. Apabila bateri digunakan (dinyahcas), tindak balas kimia berlaku di dalamnya untuk menghasilkan arus elektrik bagi menguasai peranti.



Imej janaan AI



KOMPONEN IOT

9

Komponen lain : Bateri Boleh Cas



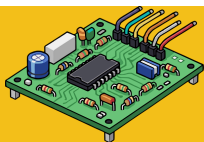
Spesifikasi

Spesifikasi	Unit	Penerangan
<u>Kapasiti</u>	mAh (miliampere-jam) atau Ah (Ampere-jam)	Ukuran jumlah tenaga yang boleh disimpan oleh bateri. Semakin tinggi nilainya, semakin lama bateri boleh menguasai peranti sebelum perlu dicas semula. <i>Contoh: Bateri 2000mAh boleh membekalkan arus 2000mA selama 1 jam, atau 200mA selama 10 jam.</i>
<u>Voltan Nominal</u>	V (Volt)	Perbezaan potensi elektrik purata semasa bateri digunakan. Setiap jenis kimia mempunyai voltan standardnya sendiri. <i>Contoh: Li-ion (3.7V), NiMH (1.2V), Asid Plumbum (2V per sel, cth: 12V).</i>
<u>Kitaran Hayat</u>	Bilangan Kitaran	Jumlah bilangan cas dan discas penuh yang boleh dilakukan sebelum kapasiti bateri merosot di bawah tahap tertentu (biasanya 80% daripada kapasiti asal). Bateri berkualiti tinggi mempunyai kitaran hayat yang lebih lama. <i>Contoh: 500 kitaran, 1000 kitaran.</i>
<u>Kadar Pelepasan Diri</u>	Peratusan per Bulan	Sejauh mana bateri kehilangan casnya sendiri apabila disimpan dan tidak digunakan. Sesetengah bateri Li-ion mempunyai kadar yang rendah, manakala NiMH jenis lama mempunyai kadar yang lebih tinggi.



Kelebihan

- 🌞 Penjimatan Kos Jangka Panjang
- 🌱 Mesra Alam
- 💪 Prestasi Lebih Konsisten
- 🚀 Kemudahan dan Kebolehgunaan



KOMPONEN IOT

10

Komponen lain : Voltage regulator

- Voltage regulator digunakan untuk menstabilkan voltan supaya sesuai dengan keperluan komponen.



Fungsi

- Tukar voltan (contoh: 7.4V → 5V / 3.3V)
- Pastikan bekalan kuasa stabil
- Lindungi komponen elektronik



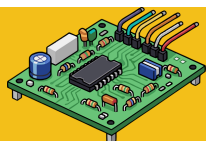
Jenis Biasa

- Linear regulator (contoh: 7805)
- Switching regulator (buck converter)



Dalam EZ Starter Kit

- 👉 Digunakan untuk:
- Bekalkan kuasa stabil kepada Pico W
- Bekalkan kuasa kepada sensor & LCD



KOMPONEN IOT

10

Komponen lain : Voltage regulator

VOLTAGE REGULATOR

Voltage regulator digunakan untuk mengekalkan voltan output yang stabil walaupun voltan input atau beban berubah.

LM2596 (Buck Converter)



IN+ IN- OUT- (3A) OUT+

7805 (Linear Regulator)



IN GND OUT (1A)

SPESIFIKASI (Contoh)

Model	Jenis	Input (V)	Output (V)	Arus (Max)
LM2596	Buck	4.5 – 40V	1.5 – 35V	3A
7805	Linear	7 – 35V	5V	1A

CARA KERJA

- **Buck Converter (LM2596):** Menurunkan voltan input kepada voltan output yang lebih rendah dengan kecekapan tinggi.
- **Linear Regulator (7805):** Menstabilkan voltan dengan membuang lebih tenaga sebagai haba.

APLIKASI



Mikropengawal



Modul Sensor

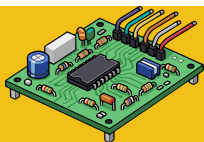


Arduino / ESP32



Elektronik DIY

Imej janaan AI



KOMPONEN IOT

11

Aksesori: Breadboard / PCB tetap

- Breadboard ialah papan ujian yang digunakan untuk membina litar elektronik tanpa pematerian.



Fungsi

- Membina dan menguji litar sementara
- Mudah ubah sambungan komponen
- Sesuai untuk pembelajaran



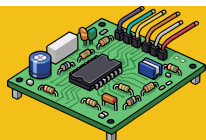
Ciri-ciri

- Mempunyai laluan sambungan dalaman
- Bahagian power rail (+ dan -)
- Lubang tersusun untuk komponen
- PCB tetap ialah papan litar yang digunakan untuk pemasangan komponen secara kekal.



Fungsi

- Menjadikan litar lebih kemas dan tersusun
- Sesuai untuk produk akhir / projek siap
- Mengurangkan risiko sambungan longgar



KOMPONEN IOT

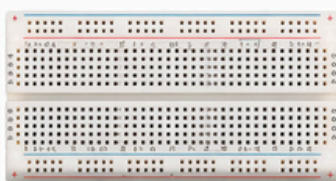
11

Aksesori: Breadboard / PCB tetap

BREADBOARD / PCB TETAP

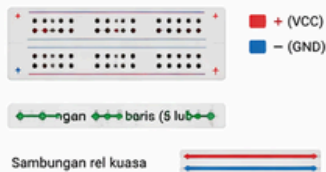
Digunakan untuk membina dan menyambung litar elektronik.
Breadboard sesuai untuk prototaip, manakala PCB tetap untuk litar kekal.

BREADBOARD (PAPAN PROTOTIP)

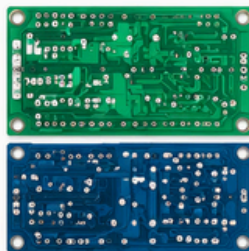


- Tidak perlu pematerian
- Lubang bersambung secara dalaman
- Sesuai untuk ujian dan prototaip litar
- Boleh diguna semula

STRUKTUR SAMBUNGAN



PCB TETAP (PRINTED CIRCUIT BOARD)



- Litar dicetak kekal
- Memerlukan pematerian
- Lebih kemas dan tahan lasak
- Sesuai untuk produk akhir

JENIS PCB



Single Layer

Double Layer

SPESIFIKASI

- Bahan: Plastik ABS, kenalan spring
- Lubang: 400 / 830 / 1660 titik (bergantung model)
- Pitch lubang: 2.54mm
- Voltan/Arus cadangan: 300V / 3A

SPESIFIKASI

- Bahan: FR-4 (Fiberglass)
- Ketebalan: 1.0mm / 1.6mm
- Salutan: HASL / ENIG
- Warna Mask: Hijau / Biru / Hitam

APLIKASI



Projek DIY



Modul Elektronik



Peralatan Rumah



Industri



SPESIFIKASI KOMPUTER



SPEKIFIKASI KOMPUTER

- Untuk membolehkan pengguna memprogram dan mengawal kit, komputer diperlukan dengan spesifikasi minimum berikut:

Perkakasan

- Laptop / PC
- Port USB
- RAM minimum: 4GB

Perisian

- Thonny IDE (MicroPython)
- atau
- Arduino IDE

Sistem Operasi

- Windows 10 / 11
- MacOS
- Linux

Lain-lain

- Kabel USB
- Driver (jika perlu)
- Sambungan Internet (untuk IoT)

Kesimpulan

- Spesifikasi teknikal penting untuk:
- Memastikan sistem berfungsi dengan betul
- Mengelakkan kerosakan komponen
- Membantu pengguna memahami sambungan dan konfigurasi

PENYEDIAAN PERISIAN



PENYEDIAAN PERISIAN

- Ini ialah proses menyediakan semua alat yang diperlukan sebelum mula menulis kod.
- 01 ➤ • Pastikan komputer mempunyai:
 - Sistem operasi (Windows / macOS / Linux)
 - Sambungan internet
 - Muat turun perisian yang diperlukan seperti IDE dan driver
 - Sediakan kabel USB untuk sambungan board

👉 Tujuan: Supaya komputer boleh berkomunikasi dengan papan mikropengawal

- 02 ➤ Pemasangan IDE (contoh: Thonny)
- IDE (Integrated Development Environment) ialah perisian untuk menulis, mengedit dan upload kod



Thonny (sesuai untuk MicroPython – Pico W)

- Muat turun Thonny di <https://thonny.org/>
- Install
- Pilih interpreter MicroPython (contoh: Raspberry Pi Pico)

👉 Tujuan: Memudahkan penulisan dan upload kod ke board

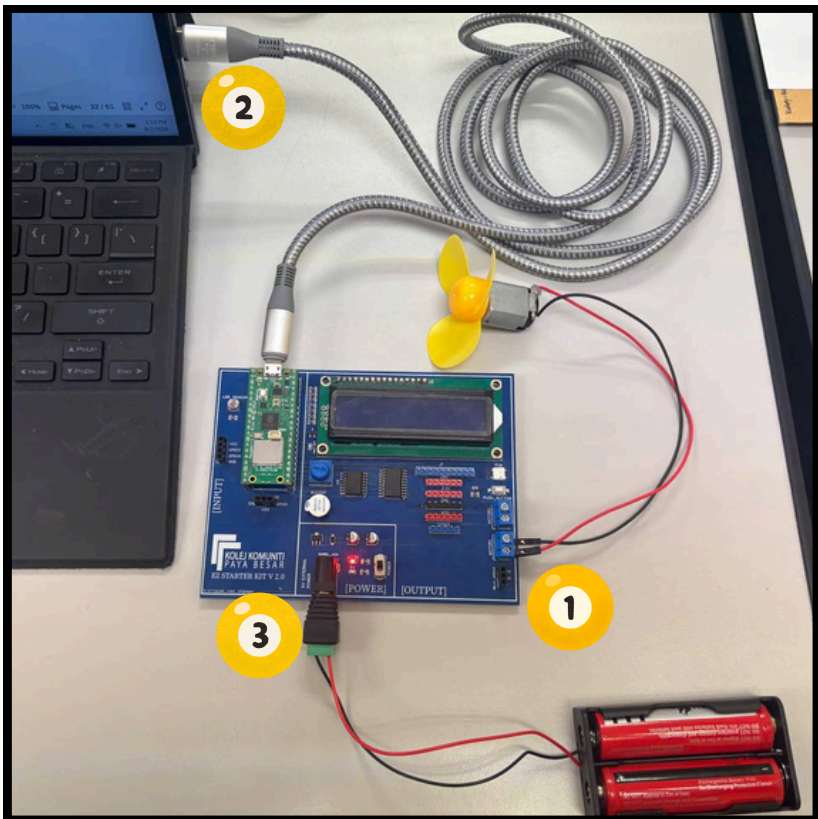


PENYEDIAAN PERISIAN

03 >>>

- 1) Setup Board (Raspberry Pi Pico W)
- 2) Sambung ke PC dengan menggunakan kabel usb micro (kabel data)
- 3) Sambungkan baterai ke Barrel Jack

👉 Tujuan: Supaya IDE kenal pasti jenis board yang digunakan

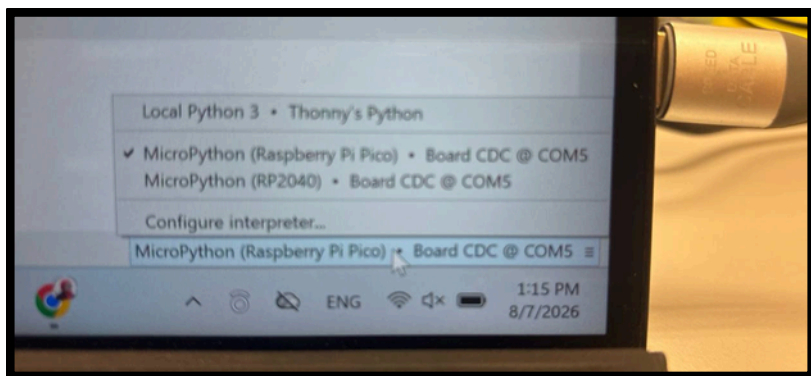




PENYEDIAAN PERISIAN

04 >>

- Buka Thonny
 - Setkan kepada MicroPython (Raspberry Pi Pico)



- Salin kod mengikut modul yang disediakan.
- Klik Run  untuk mencuba.
- Semoga berjaya!!!



MODUL AKTIVITI



MODUL AKTIVITI

ARAHAN

SILA IMBAS KOD QR DIBAWAH

- Sedia Peranti: Buka aplikasi Kamera atau mana-mana aplikasi Pengimbas QR Code pada telefon pintar / tablet anda.
- Imbas Kod QR: Halakan kamera secara tepat ke arah Kod QR Modul Aktiviti EZ Starter Kit V2 di bawah.
- Buka Pautan: Tekan pautan (URL) yang muncul pada skrin peranti anda untuk memuat turun atau melihat Kod Sumber (Source Code) bagi setiap aktiviti.
- Salin & Uji: Salin kod berkenaan ke dalam perisian Thonny IDE untuk memulakan pengaturcaraan Raspberry Pi Pico W anda.



Kod Sumber (*Source Code*) untuk semua modul



MODUL AKTIVITI



MODUL 1



LCD DAN BUZER

Di akhir modul ini, anda akan dapat:

1. Menghubungkan dan mengawal paparan LCD 16x2 menggunakan talian I2C (GP20 & GP21).
2. Mengawal Buzzer menggunakan pin Digital Output (GP7).
3. Menghasilkan maklum balas audio (bunyi buzzer) serentak dengan paparan mesej pada LCD.



Boleh salin kod dibawah ke dalam perisian Thonny, kemudian simpan fail sebagai **Lcd_Buzzer.py**

```
from machine import Pin, I2C
import time
```

```
# --- 1. CONFIG PIN (EZ STARTER KIT V2.0) ---
buzzer = Pin(7, Pin.OUT)
```

```
# Gunakan I2C ID 0 untuk GP20 (SDA) & GP21 (SCL)
i2c = I2C(0, sda=Pin(20), scl=Pin(21), freq=100000)
```

```
# Alamat I2C LCD anda
LCD_ADDR = 0x27
```

```
# --- 2. DRIVER UNTUK LCD I2C ---
def lcd_write_byte(data, mode=0):
    # mode: 0 = Command, 1 = Data
    high_nibble = data & 0xF0
    low_nibble = (data << 4) & 0xF0
    backlight = 0x08 # Backlight ON
```




MODUL AKTIVITI

MODUL 1

LCD DAN BUZER

```
# Hantar Nibble Atas
i2c.writeto(LCD_ADDR, bytes([high_nibble | mode | backlight |
0x04]))
time.sleep_us(1)
i2c.writeto(LCD_ADDR, bytes([(high_nibble | mode | backlight) &
~0x04]))

# Hantar Nibble Bawah
i2c.writeto(LCD_ADDR, bytes([low_nibble | mode | backlight |
0x04]))
time.sleep_us(1)
i2c.writeto(LCD_ADDR, bytes([(low_nibble | mode | backlight) &
~0x04]))
time.sleep_us(50)

def lcd_command(cmd):
    lcd_write_byte(cmd, mode=0)

def lcd_data(char_code):
    lcd_write_byte(char_code, mode=1)

def lcd_init():
    time.sleep_ms(50)
    lcd_command(0x33)
    lcd_command(0x32)
    lcd_command(0x28) # Mod 4-bit, 2 Baris
    lcd_command(0x0C) # Paparan ON, Cursor OFF
    lcd_command(0x01) # Clear Screen
    time.sleep_ms(5)
```



MODUL AKTIVITI

MODUL 1

LCD DAN BUZER

```
def lcd_clear():  
    lcd_command(0x01)  
    time.sleep_ms(2)  
  
def lcd_putstr(text):  
    for char in text:  
        if char == '\n':  
            lcd_command(0xC0) # Pindah ke Baris Kedua  
        else:  
            lcd_data(ord(char))  
  
# --- 3. ATUR CARA UTAMA ---  
try:  
    print("Mulakan LCD & Buzzer..")  
    lcd_init()  
  
    # Papar Teks Utama  
    lcd_clear()  
    lcd_putstr("EZ STARTER KIT\nBUZZER: ON!")  
  
    # Bunyikan Buzzer  
    buzzer.value(1)  
    time.sleep(1)  
    buzzer.value(0)  
  
    # Kemaskini Paparan Selepas Buzzer Muted  
    time.sleep(0.5)  
    lcd_clear()  
    lcd_putstr("MODUL 1: OK!\nBUZZER: OFF")
```



MODUL AKTIVITI

MODUL 1

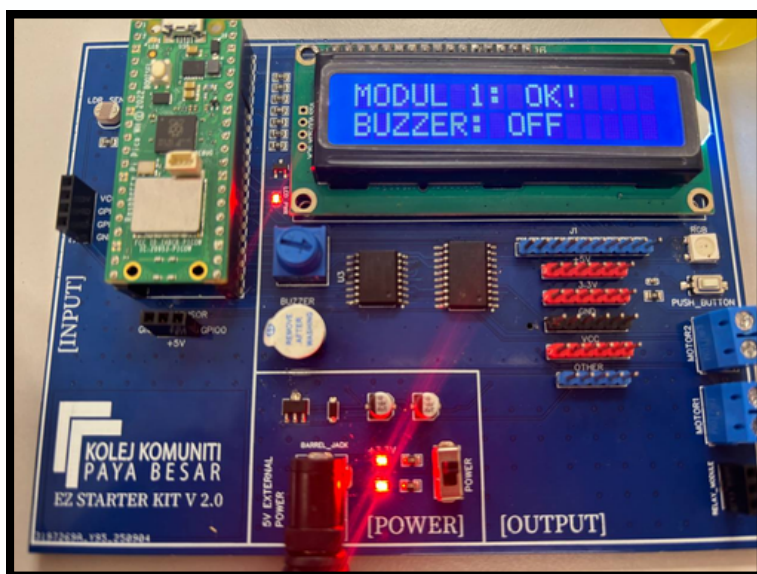
LCD DAN BUZER

```
print("Berjaya!")
```

```
except Exception as e:  
    print("Ralat berlaku:", e)
```



HASILNYA





MODUL AKTIVITI

MODUL 2

INTEGRASI PUSH BUTTON, LCD DAN BUZER

Di akhir modul ini, anda akan dapat:

1. Membaca masukan (Digital Input) daripada Push Button pada pin GP14.
2. Menggunakan Internal Pull-Down Resistor untuk memastikan bacaan butang stabil.
3. Menggabungkan logik: Apabila butang ditekan, Buzzer berbunyi dan LCD memaparkan mesej baru.



Boleh salin kod dibawah ke dalam perisian Thonny, kemudian simpan fail sebagai **Lcd_Buzzer_PushButton.py**

```
from machine import Pin, I2C
import time
```

```
# --- 1. CONFIG PIN (EZ STARTER KIT V2.0) ---
buzzer = Pin(7, Pin.OUT)
```

```
# Push Button disambungkan ke GP14 (Rujuk Skematik)
# Gunakan Pin.PULL_DOWN supaya nilai default ialah 0 (LOW)
dan 1 (HIGH) apabila ditekan
button = Pin(14, Pin.IN, Pin.PULL_DOWN)
```

```
# Gunakan I2C ID 0 untuk GP20 (SDA) & GP21 (SCL)
i2c = I2C(0, sda=Pin(20), scl=Pin(21), freq=100000)
LCD_ADDR = 0x27
```



MODUL AKTIVITI

MODUL 2

INTEGRASI PUSH BUTTON, LCD DAN BUZER

--- 2. DRIVER UNTUK LCD ---

```
def lcd_write_byte(data, mode=0):
```

```
    high_nibble = data & 0xF0
```

```
    low_nibble = (data << 4) & 0xF0
```

```
    backlight = 0x08 # Backlight ON
```

```
i2c.writeto(LCD_ADDR, bytes([high_nibble | mode | backlight |  
0x04]))
```

```
    time.sleep_us(1)
```

```
i2c.writeto(LCD_ADDR, bytes([(high_nibble | mode | backlight) &  
~0x04]))
```

```
i2c.writeto(LCD_ADDR, bytes([low_nibble | mode | backlight |  
0x04]))
```

```
    time.sleep_us(1)
```

```
i2c.writeto(LCD_ADDR, bytes([(low_nibble | mode | backlight) &  
~0x04]))
```

```
    time.sleep_us(50)
```

```
def lcd_command(cmd):
```

```
    lcd_write_byte(cmd, mode=0)
```

```
def lcd_data(char_code):
```

```
    lcd_write_byte(char_code, mode=1)
```



MODUL AKTIVITI

MODUL 2

INTEGRASI PUSH BUTTON, LCD DAN BUZER

```
def lcd_init():
    time.sleep_ms(50)
    lcd_command(0x33)
    lcd_command(0x32)
    lcd_command(0x28) # Mod 4-bit, 2 Baris
    lcd_command(0x0C) # Paparan ON, Cursor OFF
    lcd_command(0x01) # Clear Screen
    time.sleep_ms(5)

def lcd_clear():
    lcd_command(0x01)
    time.sleep_ms(2)

def lcd_putstr(text):
    for char in text:
        if char == '\n':
            lcd_command(0xC0) # Pindah ke Baris Kedua
        else:
            lcd_data(ord(char))

# --- 3. ATUR CARA UTAMA ---
try:
    print("Mula Modul 1 & Modul 2...")
    lcd_init()
```



MODUL AKTIVITI

MODUL 2

INTEGRASI PUSH BUTTON, LCD DAN BUZER

```
# -----  
# FASA 1: MODUL 1 (Ujian LCD & Buzzer)  
# -----  
lcd_clear()  
lcd_putstr("EZ STARTER KIT\nBUZZER: ON!")  
  
buzzer.value(1)  
time.sleep(1)  
buzzer.value(0)  
  
time.sleep(0.5)  
lcd_clear()  
lcd_putstr("MODUL 1: OK!\nSedia Modul 2...")  
time.sleep(2)  
  
# -----  
# FASA 2: MODUL 2 (Sedia Menunggu Push Button Ditekan)  
# -----  
lcd_clear()  
lcd_putstr("Tekan Button\npada PCB...")  
  
while True:  
    # Semak jika push button ditekan (nilai == 1)  
    if button.value() == 1:  
        # Papar mesej pada LCD  
        lcd_clear()  
        lcd_putstr("Push Button\nditekan!")
```



MODUL AKTIVITI

MODUL 2

INTEGRASI PUSH BUTTON, LCD DAN BUZER

```
# Bunyikan buzzer
buzzer.value(1)
time.sleep(0.5)
buzzer.value(0)
```

```
# Rehat sebentar supaya tidak 'double trigger'
time.sleep(1)
```

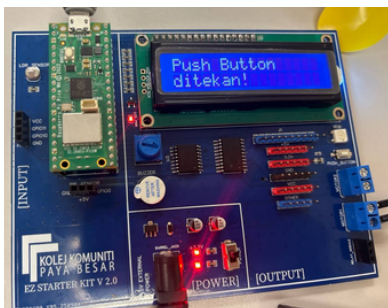
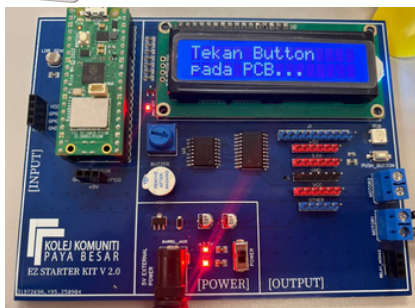
```
# Kembali ke paparan sedia
lcd_clear()
lcd_putstr("Tekan Button\npada PCB...")
```

```
time.sleep(0.05) # Delay kecil untuk kestabilan pembacaan
button
```

```
except Exception as e:
    print("Ralat berlaku:", e)
```



HASILNYA





MODUL AKTIVITI

MODUL 3

LDR SENSOR

Di akhir modul ini, anda akan dapat:

1. Membaca isyarat digital daripada LDR Sensor pada pin GP15.
2. Mengesan perubahan keadaan cahaya (Contoh: Menutup LDR dengan jari untuk mensimulasikan keadaan GELAP).
3. Mengemaskini paparan LCD dan membunyikan buzzer apabila keadaan gelap dikesan (seperti fungsi lampu jalan automatik / penggera malam).



Boleh salin kod dibawah ke dalam perisian Thonny, kemudian simpan fail sebagai **LDR_sensor.py**

```
from machine import Pin, I2C
import time
```

```
# --- 1. CONFIG PIN (EZ STARTER KIT V2.0) ---
```

```
# LDR Sensor disambungkan ke GP15 mengikut Skematik
ldr = Pin(15, Pin.IN)
```

```
# Buzzer pada GP7
buzzer = Pin(7, Pin.OUT)
```

```
# Configuration I2C untuk LCD (SDA=GP20, SCL=GP21)
i2c = I2C(0, sda=Pin(20), scl=Pin(21), freq=100000)
LCD_ADDR = 0x27
```

```
# --- 2. DRIVER UNTUK LCD I2C ---
def lcd_write_byte(data, mode=0):
    high_nibble = data & 0xF0
    low_nibble = (data << 4) & 0xF0
    backlight = 0x08 # Backlight ON
```



MODUL AKTIVITI

MODUL 3

LDR SENSOR

```
i2c.writeto(LCD_ADDR, bytes([high_nibble | mode | backlight  
| 0x04]))  
time.sleep_us(1)  
i2c.writeto(LCD_ADDR, bytes([(high_nibble | mode | backlight)  
& ~0x04]))
```

```
i2c.writeto(LCD_ADDR, bytes([low_nibble | mode | backlight |  
0x04]))  
time.sleep_us(1)  
i2c.writeto(LCD_ADDR, bytes([(low_nibble | mode | backlight)  
& ~0x04]))  
time.sleep_us(50)
```

```
def lcd_command(cmd):  
    lcd_write_byte(cmd, mode=0)
```

```
def lcd_data(char_code):  
    lcd_write_byte(char_code, mode=1)
```

```
def lcd_init():  
    time.sleep_ms(50)  
    lcd_command(0x33)  
    lcd_command(0x32)  
    lcd_command(0x28) # Mod 4-bit, 2 Baris  
    lcd_command(0x0C) # Paparan ON, Cursor OFF  
    lcd_command(0x01) # Clear Screen  
    time.sleep_ms(5)
```



MODUL AKTIVITI

MODUL 3

LDR SENSOR

```
def lcd_clear():
    lcd_command(0x01)
    time.sleep_ms(2)

def lcd_putstr(text):
    for char in text:
        if char == '\n':
            lcd_command(0xC0) # Pindah ke Baris Kedua
        else:
            lcd_data(ord(char))

# --- 3. ATUR CARA UTAMA ---
try:
    print("Mula Modul 3: LDR Sensor...")
    lcd_init()

    # Paparan Permulaan
    lcd_clear()
    lcd_putstr("Modul 3 :\nLDR Sensor")
    time.sleep(2)

    status_terakhir = -1

    while True:
        # Baca nilai LDR (0 atau 1)
        bacaan_ldr = ldr.value()
```



MODUL AKTIVITI

MODUL 3

LDR SENSOR

```
# Kemaskini paparan hanya bila status berubah
if bacaan_ldr != status_terakhir:
    if bacaan_ldr == 1:
        # Keadaan Apabila LDR Ditutup / Gelap
        lcd_clear()
        lcd_putstr("Cahaya Cukup\n(Terang)")

        # Bunyikan buzzer pendek sebagai amaran
        buzzer.value(0)
        time.sleep(0.2)
        buzzer.value(0)

    else:
        # Keadaan Normal / Terang
        lcd_clear()
        lcd_putstr("Kekurangan\nCahaya (Gelap)")
        buzzer.value(1)

    status_terakhir = bacaan_ldr

    time.sleep(0.1)

except Exception as e:
    print("Ralat berlaku:", e)
```



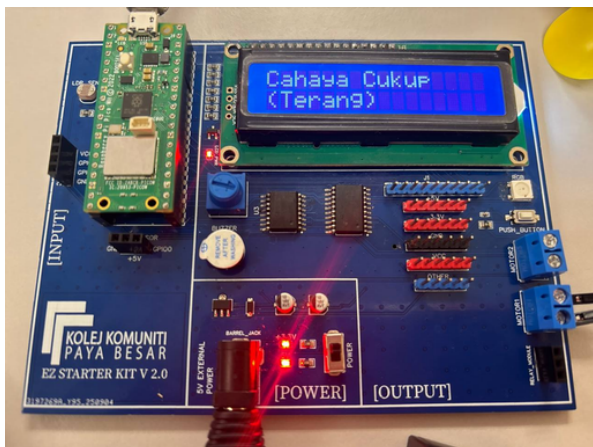
MODUL AKTIVITI

MODUL 3

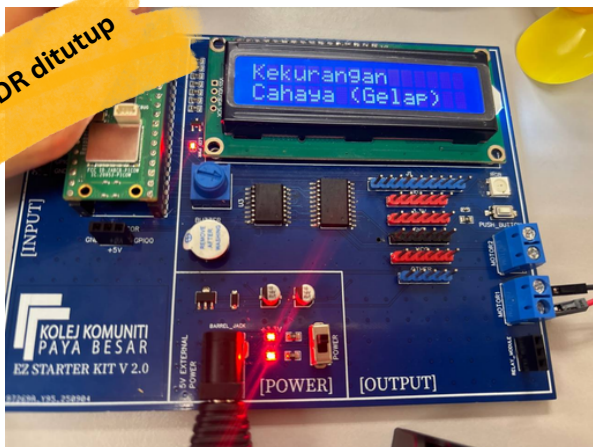
LDR SENSOR



HASILNYA



Bila LDR ditutup





MODUL AKTIVITI

MODUL 4

RGB : PUTIH, MERAH, BIRU

Di akhir modul ini, anda akan dapat:

1. Mengawal LED RGB WS2812B pada pin GP18 menggunakan modul neopixel terbina dalam MicroPython.
2. Menghasilkan warna asas: Putih (White), Merah (Red), dan Biru (Blue).
3. Mencipta perulangan automatik tukar warna setiap 5 saat serentak dengan maklumat warna pada LCD.



Boleh salin kod dibawah ke dalam perisian Thonny, kemudian simpan fail sebagai **RGB.py**

```
from machine import Pin, I2C
import neopixel
import time
```

```
# --- 1. CONFIG PIN (EZ STARTER KIT V2.0) ---
# RGB LED WS2812B disambung ke GP18 (Rujuk Skematik)
# Terdapat 1 unit LED RGB pada board (num_leds = 1)
RGB_PIN = 18
NUM_LEDS = 1
np = neopixel.NeoPixel(Pin(RGB_PIN), NUM_LEDS)

# Configuration I2C untuk LCD (SDA=GP20, SCL=GP21)
i2c = I2C(0, sda=Pin(20), scl=Pin(21), freq=100000)
LCD_ADDR = 0x27
```



MODUL AKTIVITI

MODUL 4

RGB : PUTIH, MERAH, BIRU

```
# --- 2. DRIVER UNTUK LCD I2C ---
def lcd_write_byte(data, mode=0):
    high_nibble = data & 0xF0
    low_nibble = (data << 4) & 0xF0
    backlight = 0x08 # Backlight ON

    i2c.writeto(LCD_ADDR, bytes([high_nibble | mode | backlight |
0x04]))
    time.sleep_us(1)
    i2c.writeto(LCD_ADDR, bytes([(high_nibble | mode | backlight)
& ~0x04]))

    i2c.writeto(LCD_ADDR, bytes([low_nibble | mode | backlight |
0x04]))
    time.sleep_us(1)
    i2c.writeto(LCD_ADDR, bytes([(low_nibble | mode | backlight)
& ~0x04]))
    time.sleep_us(50)

def lcd_command(cmd):
    lcd_write_byte(cmd, mode=0)

def lcd_data(char_code):
    lcd_write_byte(char_code, mode=1)
```



MODUL AKTIVITI

MODUL 4

RGB : PUTIH, MERAH, BIRU

```
def lcd_init():
    time.sleep_ms(50)
    lcd_command(0x33)
    lcd_command(0x32)
    lcd_command(0x28) # Mod 4-bit, 2 Baris
    lcd_command(0x0C) # Paparan ON, Cursor OFF
    lcd_command(0x01) # Clear Screen
    time.sleep_ms(5)
```

```
def lcd_clear():
    lcd_command(0x01)
    time.sleep_ms(2)
```

```
def lcd_putstr(text):
    for char in text:
        if char == '\n':
            lcd_command(0xC0) # Pindah ke Baris Kedua
        else:
            lcd_data(ord(char))
```

--- 3. FUNGSI SET WARNA RGB ---

Format Nilai Warna WS2812B: (Red, Green, Blue) daripada julat 0 hingga 255

WARNA_PUTIH = (255, 255, 255)

WARNA_MERAH = (255, 0, 0)

WARNA_BIRU = (0, 0, 255)

```
def tukar_warna(kod_warna):
    np[0] = kod_warna # Set warna pada LED pertama (index 0)
    np.write()        # Hantar data ke LED RGB
```




MODUL AKTIVITI

MODUL 4

RGB : PUTIH, MERAH, BIRU

--- 4. ATUR CARA UTAMA ---

try:

```
print("Mula Modul 5: RGB LED...")  
lcd_init()
```

Paparan Permulaan

```
lcd_clear()  
lcd_putstr("Modul 5 :\nRGB LED WS2812B")  
time.sleep(2)
```

while True:

1. WARNA PUTIH

```
tukar_warna(WARNA_PUTIH)  
lcd_clear()  
lcd_putstr("Modul 5: RGB\nWarna: PUTIH")  
time.sleep(5)
```

2. WARNA MERAH

```
tukar_warna(WARNA_MERAH)  
lcd_clear()  
lcd_putstr("Modul 5: RGB\nWarna: MERAH")  
time.sleep(5)
```

3. WARNA BIRU

```
tukar_warna(WARNA_BIRU)  
lcd_clear()  
lcd_putstr("Modul 5: RGB\nWarna: BIRU")  
time.sleep(5)
```

except Exception as e:

```
print("Ralat berlaku:", e)
```



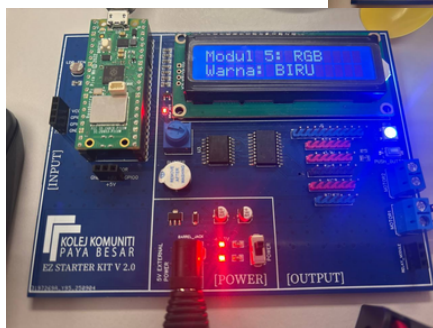
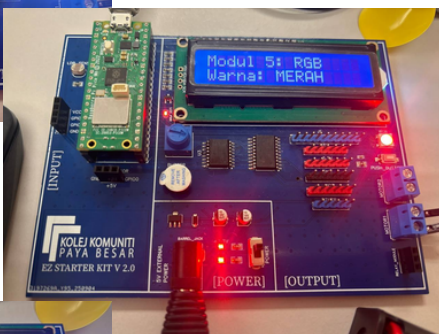
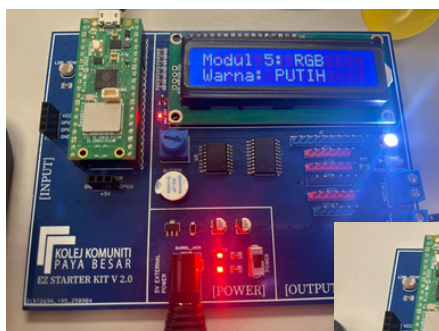
MODUL AKTIVITI

MODUL 4

RGB : PUTIH, MERAH,
BIRU



HASILNYA





MODUL AKTIVITI

MODUL 5

RGB DAN PUSH BUTTON

Di akhir modul ini, anda akan dapat:

1. Menukar mod warna LED RGB setiap kali Push Button (GP14) ditekan.
2. Mengemaskini status paparan LCD serentak dengan penukaran warna RGB.



Boleh salin kod dibawah ke dalam perisian Thonny, kemudian simpan fail sebagai **RGB_PushButton.py**

```
from machine import Pin, I2C
import neopixel
import time

# --- 1. CONFIG PIN (EZ STARTER KIT V2.0) ---
# Push Button pada GP14
button = Pin(14, Pin.IN, Pin.PULL_DOWN)

# RGB LED WS2812B pada GP18
RGB_PIN = 18
NUM_LEDS = 1
np = neopixel.NeoPixel(Pin(RGB_PIN), NUM_LEDS)

# Configuration I2C untuk LCD (SDA=GP20, SCL=GP21)
i2c = I2C(0, sda=Pin(20), scl=Pin(21), freq=100000)
LCD_ADDR = 0x27
```



MODUL AKTIVITI

MODUL 5

RGB DAN PUSH BUTTON

```
# --- 2. DRIVER UNTUK LCD I2C ---
def lcd_write_byte(data, mode=0):
    high_nibble = data & 0xF0
    low_nibble = (data << 4) & 0xF0
    backlight = 0x08 # Backlight ON

    i2c.writeto(LCD_ADDR, bytes([high_nibble | mode | backlight |
0x04]))
    time.sleep_us(1)
    i2c.writeto(LCD_ADDR, bytes([(high_nibble | mode | backlight)
& ~0x04]))

    i2c.writeto(LCD_ADDR, bytes([low_nibble | mode | backlight |
0x04]))
    time.sleep_us(1)
    i2c.writeto(LCD_ADDR, bytes([(low_nibble | mode | backlight) &
~0x04]))
    time.sleep_us(50)

def lcd_command(cmd):
    lcd_write_byte(cmd, mode=0)

def lcd_data(char_code):
    lcd_write_byte(char_code, mode=1)
```



MODUL AKTIVITI

MODUL 5

RGB DAN PUSH BUTTON

```
def lcd_init():
    time.sleep_ms(50)
    lcd_command(0x33)
    lcd_command(0x32)
    lcd_command(0x28) # Mod 4-bit, 2 Baris
    lcd_command(0x0C) # Paparan ON, Cursor OFF
    lcd_command(0x01) # Clear Screen
    time.sleep_ms(5)

def lcd_clear():
    lcd_command(0x01)
    time.sleep_ms(2)

def lcd_putstr(text):
    for char in text:
        if char == '\n':
            lcd_command(0xC0) # Pindah ke Baris Kedua
        else:
            lcd_data(ord(char))

# --- 3. SET WARNA & TETAPAN MOD ---
WARNA_PUTIH = (255, 255, 255)
WARNA_MERAH = (255, 0, 0)
WARNA_BIRU = (0, 0, 255)

def tukar_warna(kod_warna):
    np[0] = kod_warna
    np.write()
```



MODUL AKTIVITI

MODUL 5

RGB DAN PUSH BUTTON

--- 4. ATUR CARA UTAMA ---

try:

```
print("Mula Modul 6: Kawalan RGB Button...")  
lcd_init()
```

```
# Paparan Permulaan
```

```
lcd_clear()
```

```
lcd_putstr("Modul 6 :\nRGB + Button")
```

```
time.sleep(2)
```

```
# Pembolehubah untuk simpan mod warna semasa (0=PUTIH,  
1=MERAH, 2=BIRU)
```

```
mod_warna = 0
```

```
# Set warna lalai (Default) kepada PUTIH apabila power ON  
tukar_warna(WARNA_PUTIH)
```

```
lcd_clear()
```

```
lcd_putstr("Tekan Button...\nWarna: PUTIH")
```

while True:

```
# Semak jika push button ditekan
```

```
if button.value() == 1:
```

```
    # Tambah mod warna (0 -> 1 -> 2 -> 0)
```

```
    mod_warna = (mod_warna + 1) % 3
```



MODUL AKTIVITI

MODUL 5

RGB DAN PUSH BUTTON

```
# Kemaskini warna & paparan mengikut mod_warna
if mod_warna == 0:
    tukar_warna(WARNA_PUTIH)
    lcd_clear()
    lcd_putstr("Tekan Button...\nWarna: PUTIH")
elif mod_warna == 1:
    tukar_warna(WARNA_MERAH)
    lcd_clear()
    lcd_putstr("Tekan Button...\nWarna: MERAH")
elif mod_warna == 2:
    tukar_warna(WARNA_BIRU)
    lcd_clear()
    lcd_putstr("Tekan Button...\nWarna: BIRU")

# Debounce delay supaya button tak tukar banyak kali
dalam 1 tekan
time.sleep(0.4)

time.sleep(0.05)

except Exception as e:
    print("Ralat berlaku:", e)
```



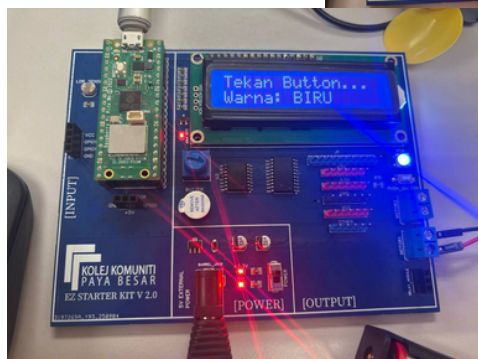
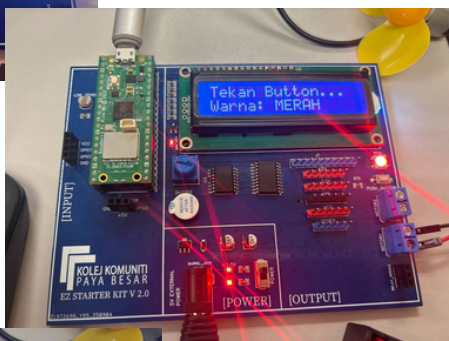
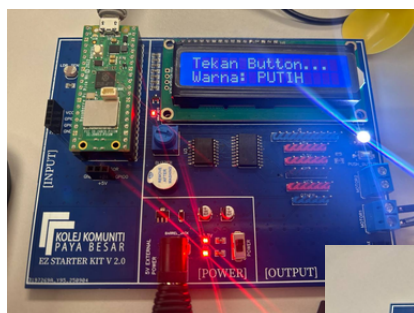
MODUL AKTIVITI

MODUL 5

RGB DAN PUSH BUTTON



HASILNYA





MODUL AKTIVITI

MODUL 6

KIPAS DAN PUSH BUTTON

Di akhir modul ini, anda akan dapat:

- LCD akan tunjuk Kipas: SoftStart / Memulakan....
- Kipas akan mula pusing secara perlahan dahulu, kemudian kelajuan dinaikkan sedikit demi sedikit.
- Ini membolehkan kipas kekal berputar tanpa menyebabkan Pico W ter-reset!



Boleh salin kod dibawah ke dalam perisian Thonny, kemudian simpan fail sebagai **Kipas_PushButton.py**

```
from machine import Pin, PWM, I2C, reset_cause,
PWRON_RESET
import time
```

```
# --- 1. CONFIG PIN (EZ STARTER KIT V2.0) ---
button = Pin(14, Pin.IN, Pin.PULL_DOWN)
```

```
# Motor 1 Pins (FAN1)
m1_in1 = Pin(2, Pin.OUT)
m1_in2 = Pin(6, Pin.OUT)
m1_en = PWM(Pin(1))
m1_en.freq(1000)
```

```
# Configuration I2C untuk LCD
i2c = I2C(0, sda=Pin(20), scl=Pin(21), freq=100000)
LCD_ADDR = 0x27
```



MODUL AKTIVITI

MODUL 6

KIPAS DAN PUSH BUTTON

```
# --- 2. DRIVER UNTUK LCD I2C ---
def lcd_write_byte(data, mode=0):
    high_nibble = data & 0xF0
    low_nibble = (data << 4) & 0xF0
    backlight = 0x08

    i2c.writeto(LCD_ADDR, bytes([high_nibble | mode | backlight |
0x04]))
    time.sleep_us(1)
    i2c.writeto(LCD_ADDR, bytes([(high_nibble | mode | backlight)
& ~0x04]))

    i2c.writeto(LCD_ADDR, bytes([low_nibble | mode | backlight |
0x04]))
    time.sleep_us(1)
    i2c.writeto(LCD_ADDR, bytes([(low_nibble | mode | backlight) &
~0x04]))
    time.sleep_us(50)

def lcd_command(cmd): lcd_write_byte(cmd, mode=0)
def lcd_data(char_code): lcd_write_byte(char_code, mode=1)
def lcd_init():
    time.sleep_ms(50)
    lcd_command(0x33)
    lcd_command(0x32)
    lcd_command(0x28)
    lcd_command(0x0C)
    lcd_command(0x01)
    time.sleep_ms(5)
```



MODUL AKTIVITI

MODUL 6

KIPAS DAN PUSH BUTTON

```
def lcd_clear(): lcd_command(0x01); time.sleep_ms(2)
def lcd_putstr(text):
    for char in text:
        if char == '\n': lcd_command(0xC0)
        else: lcd_data(ord(char))
```

```
def kipas_off():
    m1_en.duty_u16(0)
    m1_in1.value(0)
    m1_in2.value(0)
```

--- 3. MENGESAN SEBAB RESET SYSTEM ---

```
lcd_init()
kipas_off()
```

```
sebab_reset = reset_cause()
```

```
if sebab_reset != PWRON_RESET:
    lcd_clear()
    lcd_putstr("Bekalan Kuasa\nTidak Mencukupi")
    time.sleep(3)
```

```
lcd_clear()
lcd_putstr("EZ Starter Kit\nVersion 2.0")
time.sleep(3)
```



MODUL AKTIVITI

MODUL 6

KIPAS DAN PUSH BUTTON

```
# --- 4. PROGRAM UTAMA MODUL 8 (SOFT-START) ---
try:
    status_kipas = False

    lcd_clear()
    lcd_putstr("Modul 6: Kipas dengan Push Button\nTekan
    Button...")

    while True:
        if button.value() == 1:
            status_kipas = not status_kipas

            if status_kipas:
                lcd_clear()
                lcd_putstr("Kipas: SoftStart\nMemulakan...")

                m1_in1.value(1)
                m1_in2.value(0)

                # SOFT-START: Naikkan kelajuan secara perlahan dari 0
                ke 40% (25000)
                # Langkah ini menghalang "Voltage Drop" mengejut!
                for duty in range(0, 25000, 1000):
                    m1_en.duty_u16(duty)
                    time.sleep(0.05)
```



MODUL AKTIVITI



MODUL 6



KIPAS DAN PUSH BUTTON

```
lcd_clear()
lcd_putstr("Kipas: RUNNING\nStatus: ON")
else:
    kipas_off()
    lcd_clear()
    lcd_putstr("Modul 6: Kipas\nStatus: STOPPED")

time.sleep(0.5)

time.sleep(0.05)

except Exception as e:
    kipas_off()
    print("Ralat:", e)
```



HASILNYA





MODUL AKTIVITI

MODUL 7

LDR CAHAYA & SENSOR SUHU DALAM PICO W

Di akhir modul ini, anda akan dapat:

- Membaca status Cahaya daripada LDR Sensor pada GP15.
- Membaca Suhu Persekitaran/Cip menggunakan ADC Internal Sensor (Channel 4) pada Raspberry Pi Pico W.
- Memaparkan nilai Suhu (°C) dan Status Cahaya secara serentak pada LCD Display.



Boleh salin kod dibawah ke dalam perisian Thonny, kemudian simpan fail sebagai **LDR_Suhu_Gelap_Terang.py**

```
from machine import Pin, ADC, I2C
import time
```

```
# --- 1. CONFIG PIN (EZ STARTER KIT V2.0) ---
# LDR Sensor pada GP15
ldr = Pin(15, Pin.IN)
```

```
# Sensor Suhu Dalaman Raspberry Pi Pico W (ADC Channel 4)
adc_suhu = ADC(4)
```

```
# Configuration I2C untuk LCD
i2c = I2C(0, sda=Pin(20), scl=Pin(21), freq=100000)
LCD_ADDR = 0x27
```



MODUL AKTIVITI

MODUL 7

LDR CAHAYA & SENSOR SUHU DALAM PICO W

```
# --- 2. DRIVER UNTUK LCD I2C ---
def lcd_write_byte(data, mode=0):
    high_nibble = data & 0xF0
    low_nibble = (data << 4) & 0xF0
    backlight = 0x08

    i2c.writeto(LCD_ADDR, bytes([high_nibble | mode | backlight |
0x04]))
    time.sleep_us(1)
    i2c.writeto(LCD_ADDR, bytes([(high_nibble | mode | backlight)
& ~0x04]))

    i2c.writeto(LCD_ADDR, bytes([low_nibble | mode | backlight |
0x04]))
    time.sleep_us(1)
    i2c.writeto(LCD_ADDR, bytes([(low_nibble | mode | backlight) &
~0x04]))
    time.sleep_us(50)

def lcd_command(cmd): lcd_write_byte(cmd, mode=0)
def lcd_data(char_code): lcd_write_byte(char_code, mode=1)
def lcd_init():
    time.sleep_ms(50)
    lcd_command(0x33); lcd_command(0x32);
    lcd_command(0x28)
    lcd_command(0x0C); lcd_command(0x01); time.sleep_ms(5)
```



MODUL AKTIVITI

MODUL 7

LDR CAHAYA & SENSOR SUHU DALAM PICO W

```
def lcd_clear(): lcd_command(0x01); time.sleep_ms(2)
def lcd_putstr(text):
    for char in text:
        if char == '\n': lcd_command(0xC0)
        else: lcd_data(ord(char))

# --- 3. FUNGSI KIRA SUHU ---
def baca_suhu_pico():
    bacaan_raw = adc_suhu.read_u16()
    voltan = bacaan_raw * (3.3 / 65535)
    suhu_celsius = 27 - (voltan - 0.706) / 0.001721
    return round(suhu_celsius, 1)

# --- 4. ATUR CARA UTAMA ---
try:
    print("Mula Modul 7: LDR & Suhu Pico (Logik Pembetulan)...")
    lcd_init()

    lcd_clear()
    lcd_putstr("Modul 7 :\nLDR & Temp Pico")
    time.sleep(2)
```




MODUL AKTIVITI

MODUL 7

LDR CAHAYA & SENSOR SUHU DALAM PICO W

```
while True:
    val_ldr = ldr.value()
    suhu = baca_suhu_pico()

    lcd_clear()
    # Baris 1: Suhu
    lcd_putstr("Suhu : " + str(suhu) + " C")

    # Baris 2: Status Cahaya (Tukar Logik: 0 = TERANG, 1 =
    GELAP)
    lcd_command(0xC0)
    if val_ldr == 0:
        lcd_putstr("Cahaya: GELAP")
    else:
        lcd_putstr("Cahaya: TERANG ")

    time.sleep(0.5)

except Exception as e:
    print("Ralat berlaku:", e)
```



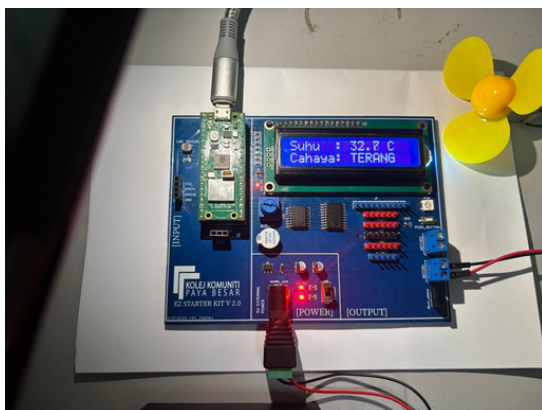
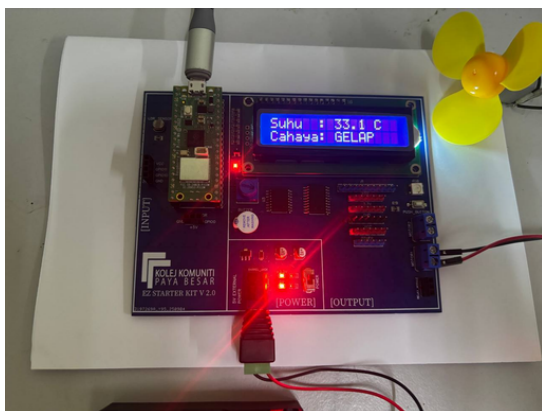
MODUL AKTIVITI

MODUL 7

LDR CAHAYA & SENSOR SUHU DALAM PICO W



HASILNYA





MODUL AKTIVITI

MODUL 8

ULTRASONIC SENSOR + LCD DISPLAY

Di akhir modul ini, anda akan dapat:

- Membaca isyarat daripada sensor ultrasonik (HC-SR04) dan memaparkan keputusan pada LCD I2C.
- Mengira jarak objek secara automatik berdasarkan selang masa penerimaan pantulan gelombang bunyi.
- Mengintegrasikan paparan antaramuka LCD 16x2 untuk menunjukkan mesej amaran "Ada Halangan!!!" dan nilai jarak semasa secara dinamik.



Boleh salin kod dibawah ke dalam perisian Thonny, kemudian simpan fail sebagai **Ultrasonic_LCD.py**

```
import time
from machine import I2C, Pin
```

```
# --- Konfigurasi LCD I2C ---
```

```
# Menetapkan I2C1 pada pin GP20 (SDA) & GP21 (SCL) mengikut skematik
```

```
i2c = I2C(0, sda=Pin(20), scl=Pin(21), freq=400000)
```

```
# Alamat default PCF8574 (biasanya 0x27 atau 0x3F)
LCD_ADDR = 0x27
```



MODUL AKTIVITI

MODUL 8

ULTRASONIC SENSOR + LCD DISPLAY

```
def lcd_send_byte(cmd, mode):  
    # Fungsionaliti menghantar arahan/data ke LCD  
    high = mode | (cmd & 0xF0) | 0x08  
    low = mode | ((cmd << 4) & 0xF0) | 0x08  
    i2c.writeto(LCD_ADDR, bytes([high, high | 0x04, high]))  
    i2c.writeto(LCD_ADDR, bytes([low, low | 0x04, low]))  
  
def lcd_init():  
    for cmd in [0x33, 0x32, 0x28, 0x0C, 0x01, 0x06]:  
        lcd_send_byte(cmd, 0)  
        time.sleep_ms(5)  
  
def lcd_putstr(text, line=0):  
    addr = 0x80 if line == 0 else 0xC0  
    lcd_send_byte(addr, 0)  
    for char in text:  
        lcd_send_byte(ord(char), 1)  
  
# --- Konfigurasi Ultrasonic Sensor ---  
# GP10 dan GP11 mengikut skematik ULTRASONIC & PCB Board  
trig = Pin(11, Pin.OUT)  
echo = Pin(10, Pin.IN)  
  
def get_distance():  
    trig.value(0)  
    time.sleep_us(2)  
    trig.value(1)  
    time.sleep_us(10)  
    trig.value(0)
```



MODUL AKTIVITI

MODUL 8

ULTRASONIC SENSOR + LCD DISPLAY

```
# Ukur durasi denyutan
while echo.value() == 0:
    signaloff = time.ticks_us()
while echo.value() == 1:
    signalon = time.ticks_us()

timepassed = signalon - signaloff
distance = (timepassed * 0.0343) / 2
return distance
```

```
# --- Tetapan Jarak Halangan (dalam cm) ---
HAD_JARAK = 10.0 # Anda boleh ubah jarak ini mengikut
kesesuaian
```

```
# --- Program Utama ---
try:
    lcd_init()
    lcd_putstr("MODUL 8:ULTRASONIC SENSOR", 0)
    lcd_putstr("BERSEDIA....", 1)
    time.sleep(2)
```

```
while True:
    dist = get_distance()

    # Semak jika terdapat halangan dalam had jarak
    if dist <= HAD_JARAK:
        lcd_putstr("Ada Halangan!!! ", 0)
        lcd_putstr("Jarak: {:.1f} cm ".format(dist), 1)
```



MODUL AKTIVITI

MODUL 8

ULTRASONIC SENSOR + LCD DISPLAY

else:

```
lcd_putstr("Tiada Halangan ", 0)
```

```
lcd_putstr("Jarak: {:.1f} cm ".format(dist), 1)
```

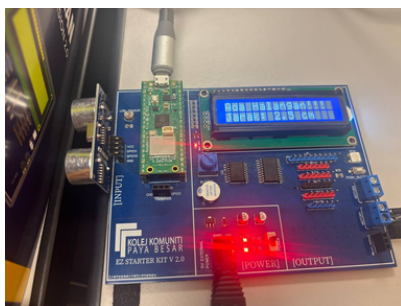
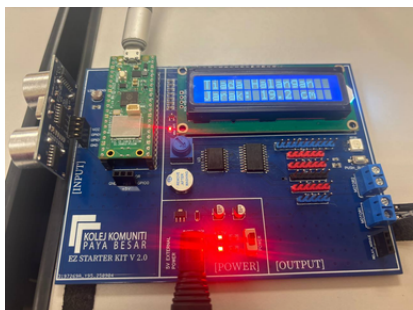
```
time.sleep(0.3)
```

except Exception as e:

```
print("Ralat:", e)
```



HASILNYA





**SEKIAN ,
TERIMA KASIH**

MANUAL **PENGUNAAN**

EZ STARTER KIT VERSION 2.0



KOLEJ KOMUNITI PAYA BESAR