TECH KIDZ AFRICA

Empowering future innovation

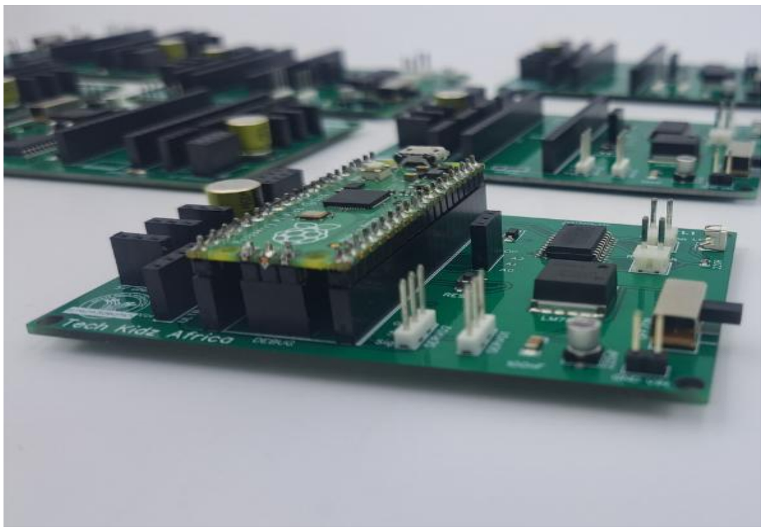
ELIMU PI ROBOTICS

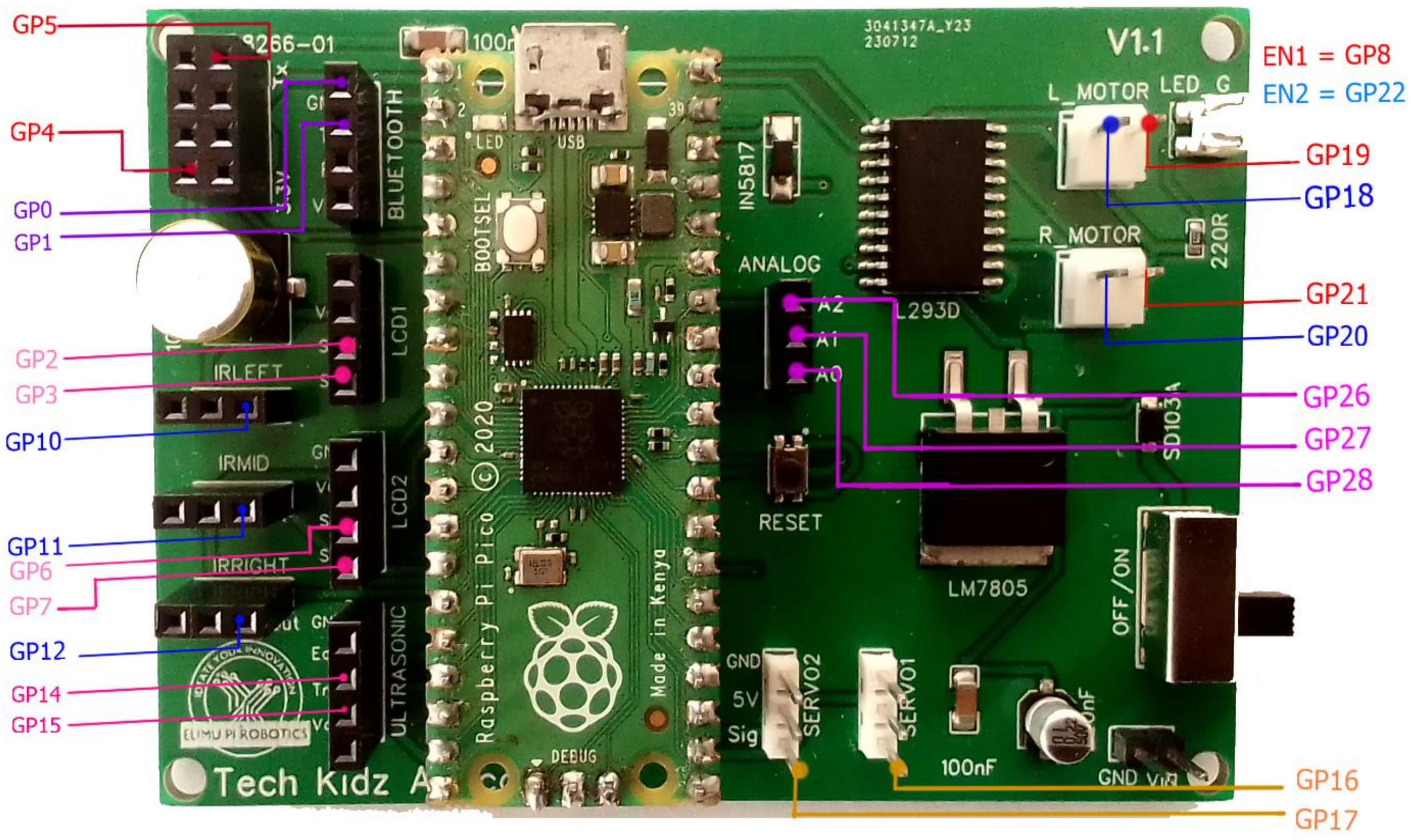
Projects Guide

This **Elimu Pi Projects' Guide** is a guidebook for robotics enthusiasts who want to learn how to design, build and program their own robotic project using Raspberry Pi. The book covers the basics of robotics, such as sensors, motors, controllers and communication and also provides step-by-step instructions for several projects. Whether you are a beginner or an expert, Elimu Pi will help you unleash your creativity and innovation in robotics.

About

The Elimu Pi Robotics is a customized programmable PCB designed circuit board that uses the Elimu Pi Pico and Raspberry Pico Microcontroller to achieve robotics and Internet of Thing(IOT) prototypes and innovations.





Catalog

Getting Started

To use the Elimu Pi Microcontroller, you first need to download the MicroPython firmware Raspberry Pi Pico.

You will require an Elimu Pi Board, Micro USB, and Thonny software for these projects.

1. Install MicroPython Firmware

To download the Firmware, visit the official MicroPython website and search “MicroPython firmware for Raspberry Pi Pico”.

2. Put Pico in Boot Mode

To put your Pico in boot mode (programming mode), press and hold the BOOTSEL button while connecting it to your computer using a micro USB cable.

3. Installing the Firmware

Once the Pico is in boot mode, it should appear as a USB mass storage device on your computer.

Copy the downloaded MicroPython firmware (.uf2 file) and paste it onto Pico's USB mass storage device. This will flash the firmware onto the Pico.

4. Restart the Pico

After flashing the firmware, disconnect and reconnect the Pico from your computer. It will now run the MicroPython firmware.

Please note

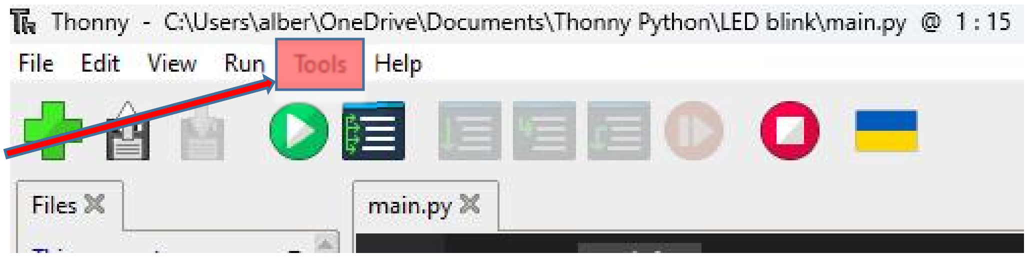
If you want your program to run without the Pi Pico being connected to the computer, you have to rename the code to “*main.py* “, and then save it to Pi Pico’s memory rather than the computer.

5. Installing Thonny IDE

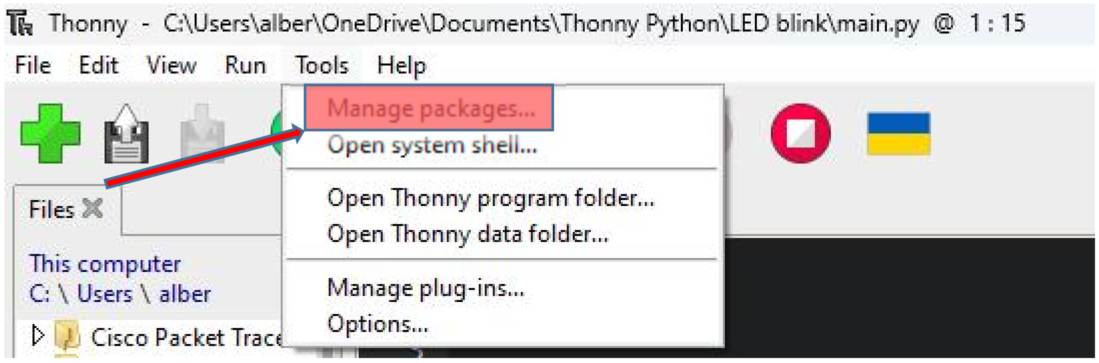
Make sure you are connected to the internet and browse the https://thonny.org/ website.
For Windows users, download the Windows x64 setup file.
After downloading, run the setup file to install.

6. Installing Libraries in Thonny

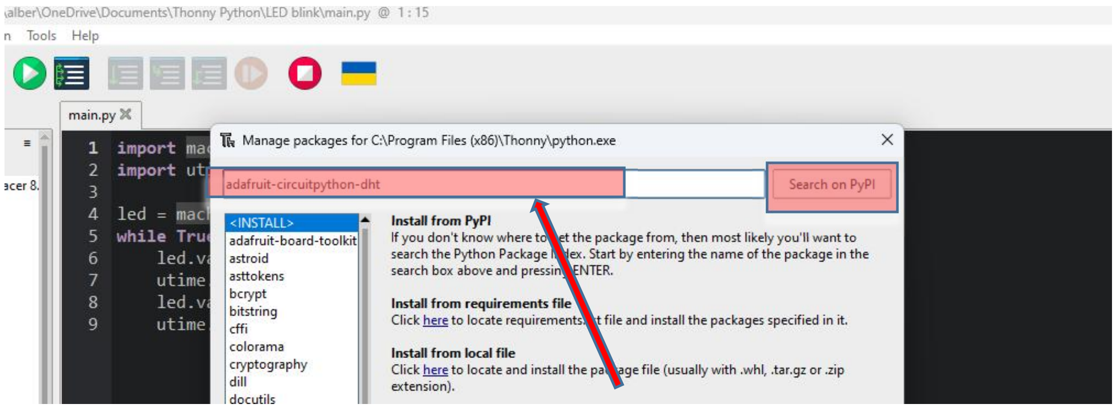
Step 1: Navigate to the **tools** panel.


Thonny - C:\Users\alber\OneDrive\Documents\Thonny Python\LED blink\main.py @ 1 : 15
File
Edit
View
Run
Tools
Help
Files
main.py

Step 2: Click the **Manage packages** in the drop-down menu.



Step 3: In the search dialogue, type in the words **adafruit-circuitpython-dht** and then click the **search on Pypl**


\alber\OneDrive\Documents\Thonny Python\LED blink\main.py @ 1 : 15
Tools
Help
main.py
Manage packages for C:\Program Files (x86)\Thonny\python.exe
adafruit-circuitpython-dht
Search on PyPI
<INSTALL>
adafruit-board-toolkit
astroid
asttokens
bcrypt
bitstring
cffi
colorama
cryptography
dill
docutils

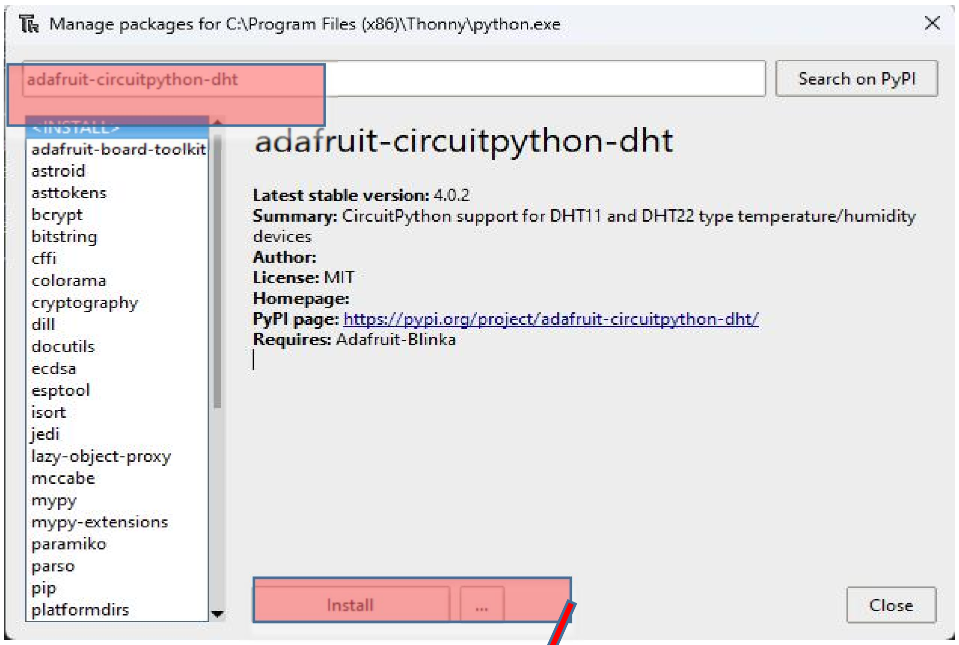
```
import ma
import ut

led = mac
while Tru
    led.va
    utime
    led.va
    utime
```

Install from PyPI
If you don't know where to get the package from, then most likely you'll want to search the Python Package Index. Start by entering the name of the package in the search box above and pressing ENTER.

Install from requirements file
Click here to locate requirements.txt file and install the packages specified in it.

Install from local file
Click here to locate and install the package file (usually with .whl, .tar.gz or .zip extension).

Step 4: Install the library as shown below.



Manage packages for C:\Program Files (x86)\Thonny\python.exe

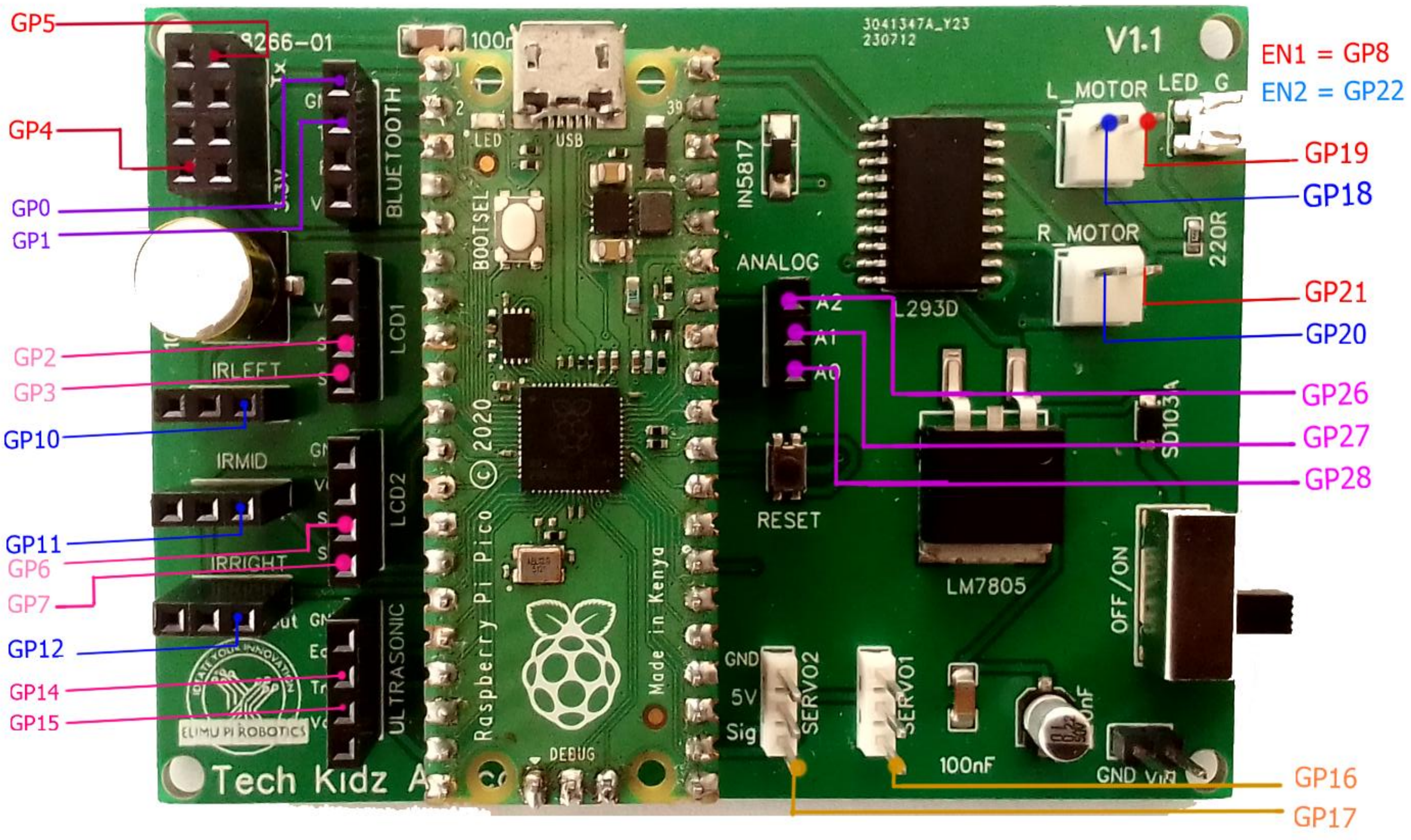
adafruit-circuitpython-dht

adafruit-circuitpython-dht

Latest stable version: 4.0.2
Summary: CircuitPython support for DHT11 and DHT22 type temperature/humidity devices
Author:
License: MIT
Homepage:
PyPI page: https://pypi.org/project/adafruit-circuitpython-dht/
Requires: Adafruit-Blinka

Other Libraries that require to be installed include;

Module	**Library to be installed**
LCD i2C (16x2)	**lcd_api.py"** and "**pico_i2c_lcd.py**
DHT11 sensor module	***dhtsensor.py***
BMP280 sensor module	***"bmp280"*** and "***bmp-280"***



Projects

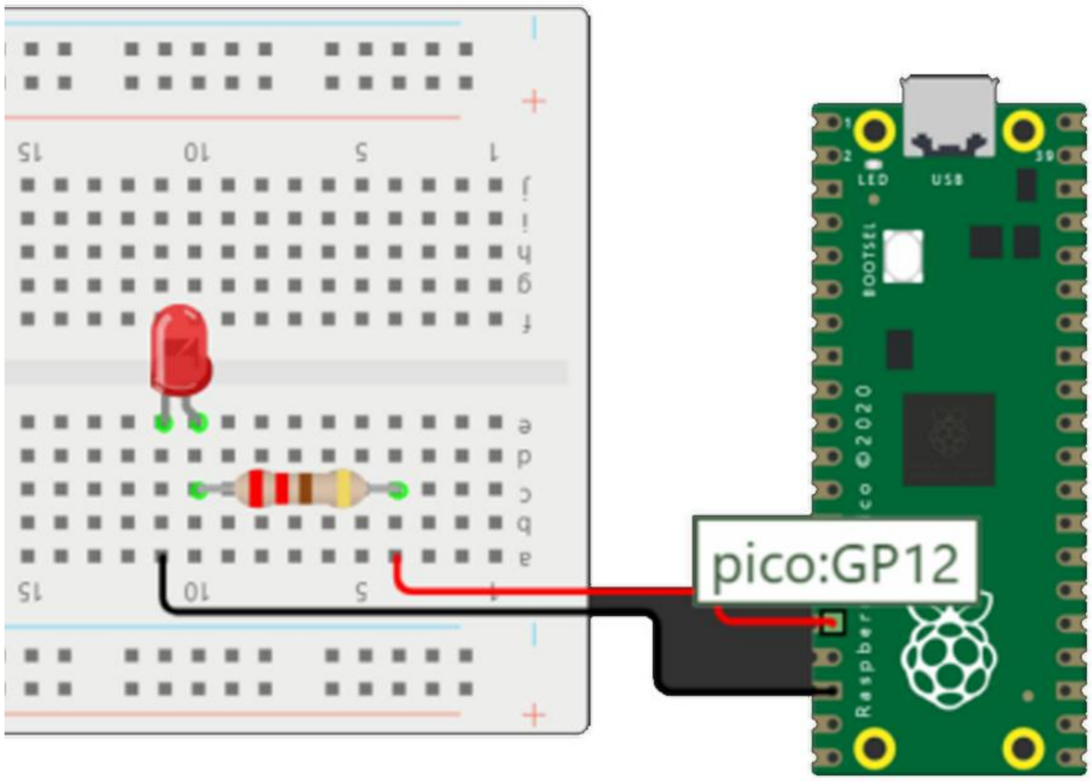
1) LED Blinking

In this project, we will write a program in Thonny to blink the Raspberry Pi Pico's on-board LED. This on-board LED is connected to **GP25**.

Procedure 1: On-Board LED

a) Plug your Pico into your computer using a micro USB cable.
b) Create a new project in Thonny and write the code below.
c) Save it as "*Blinking.py*"
d) Click the "Run" button in Thonny to execute the code.

```
import machine
import utime
led = machine.Pin(25, machine.Pin.OUT) # Set GPIO 25 as an output pin
while True:
    led.value(1)         #Turns the LED on
    utime.sleep(1)       # Keep the LED on for 1 second
    led.value(0)         # Turn off the LED
    utime.sleep(0.5)     # Keep the LED off for 0.5 seconds
```



Procedure 2: Connecting an external LED

In this project, we will replace the on-board pin connection from **GP25** to the physical **GP12** pin.

Requirements

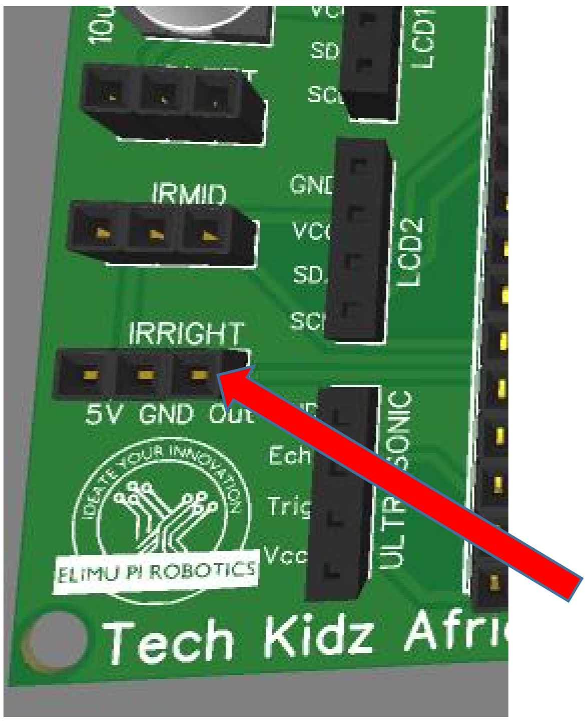
- LED any color
- 220R Ohms resistor
- Breadboard
- Jumper wires

a) Connect the LED's longer leg (anode) to one end of the resistor.
b) Connect the other end of the resistor to the "***OUT***" pin of "***IR-RIGHT***". (This OUT pin is connected to GP12 of the Pi Pico).
c) Connect the short leg (cathode) of the LED to one of the ground (GND) pins on the Pico.
d) Using the "***Blinking.py***" Program, modify the LED's pin from GPIO25 to GPIO12.
e) The new line in the code should be as follows:

```
led = machine.Pin(12, machine.Pin.OUT)
```

f) Click the "Run" button in Thonny to execute the code.

Save the program in the Pi Pico's memory as "**main.py**". This will run the code without necessarily connecting the board to your PC.



IR_RIGHT OUT
Connected to GP12

2) Traffic Lights Control

In this project, we will create traffic control lighting using LEDs and Resistors. It helps the learner acquire skills in controlling multiple outputs using the Elimu Pi Programming Board.

Requirements

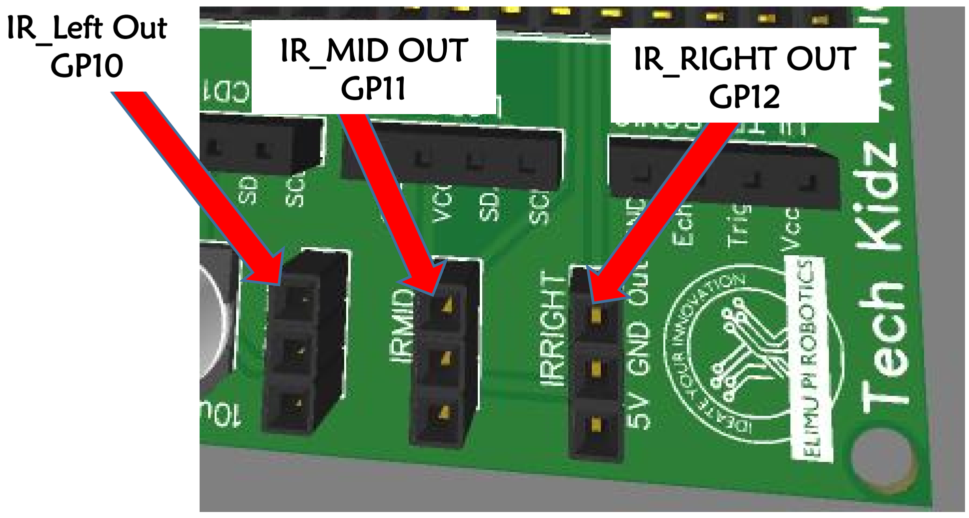
- Breadboard
- 3 LEDs (Red, Yellow, Green)
- 3 Resistors, 220R ohms each
- Jumper wires

Procedure

a) Connect the longer leg (anode) of the red LED to one end of resistor1.
b) Connect the other end of resistor1 to the "*OUT*" pin of "*IR_RIGHT*". (This OUT pin is connected to **GP12** of the Pi Pico).
c) Connect the longer leg (anode) of the yellow LED to one end of resistor 2.
d) Connect the other end of resistor 2 to the "*OUT*" pin of "*IR_MID*". (This OUT pin is connected to **GP11** of the Pi Pico).
e) Connect the longer leg (anode) of the green LED to one end of the resistor3.
f) Connect the other end of resistor 3 to the "*OUT*" pin of "*IR_LEFT*". (This OUT pin is connected to **GP10** of the Pi Pico).
g) Connect the short legs (cathode) of all the LEDs to a common ground (GND) pin on the Pico.

Working Principles

The circuit starts by turning on the red LED for 3 seconds then goes off. The yellow LED turns on for 1 second then the green LED turns on for 3 seconds. After the green LED goes off, the yellow LED turns on for 1 second followed by the red one.
The cycle repeats from **RED → YELLOW → GREEN → YELLOW → RED.**



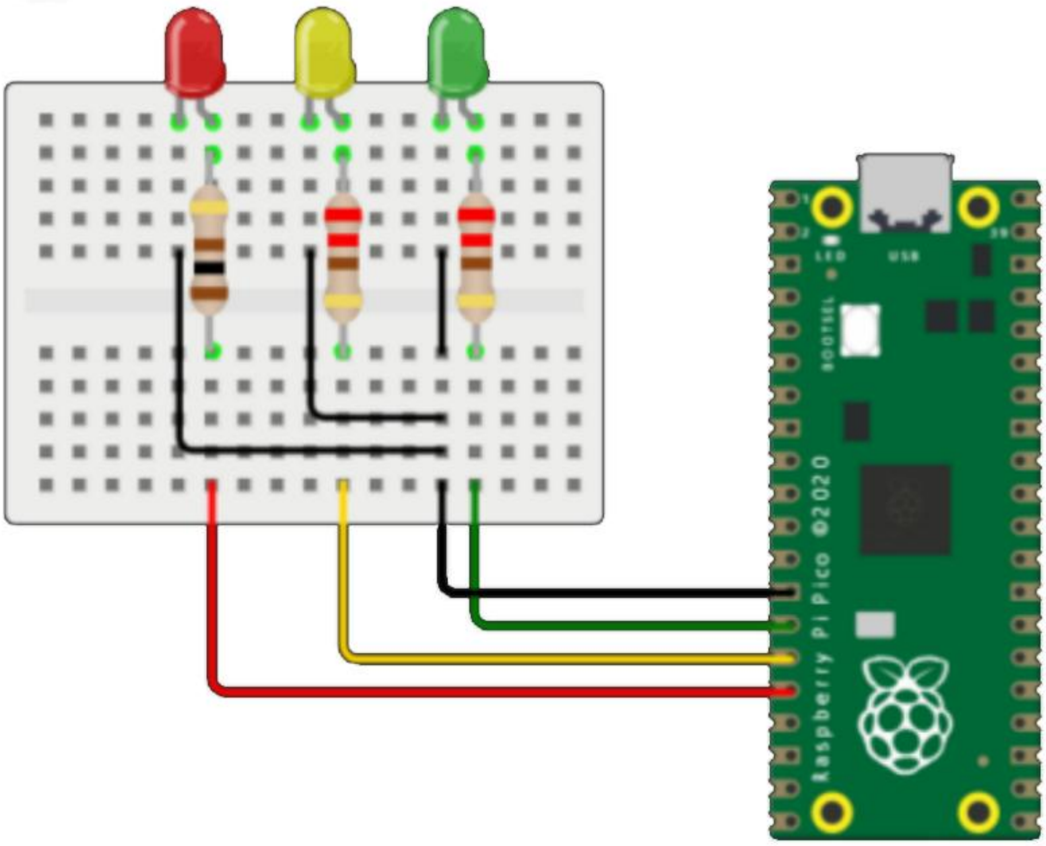


Figure 1: Traffic Light Diagram

```
import machine
import utime
red = machine.Pin(12, machine.Pin.OUT) # Set GPIO 25 as an output pin
yellow = machine.Pin(11, machine.Pin.OUT)
green = machine.Pin(10, machine.Pin.OUT)
while True:
    red.value(1)      #Turns the red LED on
    utime.sleep(3)    # Keep the LED on for 1 second
    red.value(0)      # Turn off the LED

    yellow.value(1) #Turns the yellow red LED on
    utime.sleep(1)    # Keep the LED on for 1 second
    yellow.value(0) # Turn off the LED

    green.value(1)    #Turns the green LED on
    utime.sleep(3)    # Keep the LED on for 1 second
    green.value(0)    # Turn off the LED

    yellow.value(1) #Turns the yellow LED on
    utime.sleep(1)    # Keep the LED on for 1 second
    yellow.value(0) # Turn off the LED
```

3) Controlling a Servo Motor

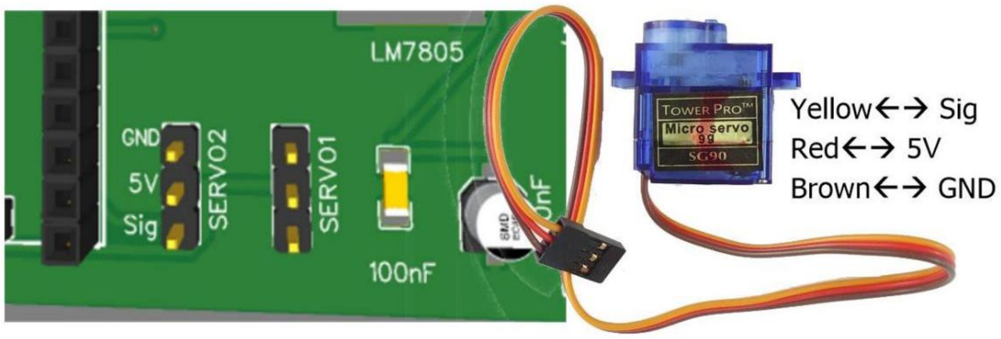
Servo motors are used to make precise angular rotations of the shaft and are commonly applied in robotics and automation. This procedure guides on how to control the Elimu Pi servo motors through predefined rotation angles.

Requirements

- Servo motor
- Jumper wires

Procedure

a) Connect the servo motor pins in the "Servo1" slot as follows; **Brown wire ←→ GND, Red wire ←→ 5V, Orange/Yellow ←→ Sig (GPIO 16.)**
b) To use the "Sevo2" slot, connect the servo motor as follows, **Brown wire ←→ GND, Red wire ←→ 5V, Orange/Yellow ←→ Sig (GPIO 17).**
c) Copy the code below in Thonny IDE then "**Run the script**" green button.
d) You can use both Servo1 and Servo2 slots at the same time, but you need to specify that in the code.



Operation

This code rotates the servo motor by 120 degrees.
To change this angle, edit the line in the code from the "**desired_angle = 120** "
to your desired angle between 0 and 180 degrees.

```
from machine import Pin, PWM
import utime

def set_servo_angle(angle, pwm):
    duty = int((angle / 180) * 8000 + 1000)
    # Adjust these values based on your servo
    pwm.duty_u16(duty)

# Define the servo control pin
servo_pin = Pin(16)
# Create a PWM object for controlling the servo
servo_pwm = PWM(servo_pin)
servo_pwm.freq(50)
desired_angle = 120 #Set the desired angle 0 to 180
set_servo_angle(desired_angle, servo_pwm)
utime.sleep(2)   # Allow time for the servo to move
```

4) LCD 16x2 Guide

This is a guide on how to use the 16x2 LCD to display names, letters, and numbers in any project. This project assumes the LCD used has the i2C adapter module and that the

Requirements

- 16x2 LCD with i2C module.

Procedure

a) The LCD 16x2 i2C module has 4 pins; GND, VCC, SDA, SCL.
b) The Elimu Pi Board has 2 slots for the LCD, namely LCD1 and LCD2.
c) The pins are connected as follows

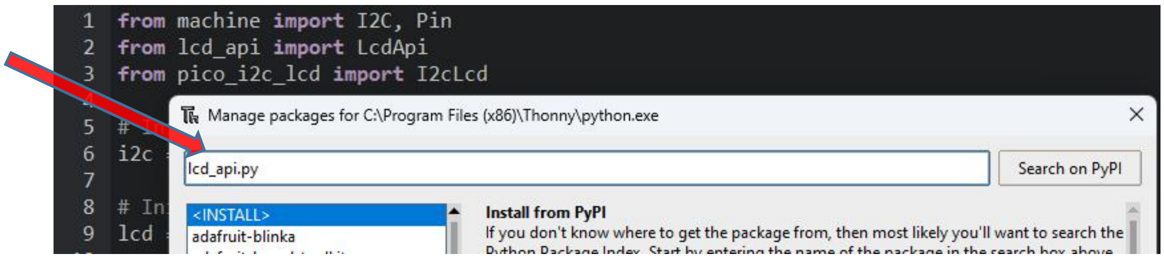
LCD1

LCD pin	Raspberry Pi
SDA	GP2
SCL	GP3

LCD2

LCD pin	Raspberry Pi
SDA	GP6
SCL	GP7

d) The GND and VCC/5V are connected to the power supply's GND and regulated 5Volts respectively.
e) Insert the LCD i2C's terminals into the LCD1 slot.
f) If you don't need to insert the LCD pins, then use jumper wires and connect the LCD to the LCD1 slot terminals.
g) Connect **SDA ←→SDA, SCL ←→SCL, 5V ←→VCC, GND ←→GND.**
h) You can use the LCD2 slot as well, but be sure to define in the code the GPIO pins used.
i) Install "**lcd_api.py**" and "**pico_i2c_lcd.py** "drivers in order to use the LCD i2C functions.



```
from machine import I2C, Pin
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd

# I
i2c

# In
lcd
```

Manage packages for C:\Program Files (x86)\Thonny\python.exe

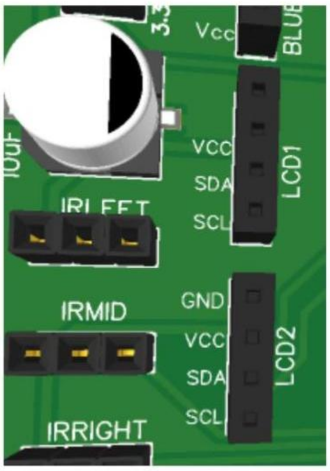
lcd_api.py

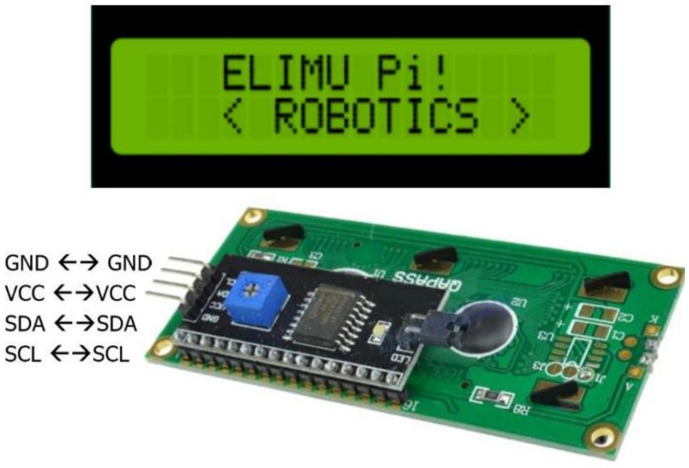
Install from PyPI

If you don't know where to get the package from, then most likely you'll want to search the

Operation

This program is used to display specified words on the 16x2 LCD.
The display has two lines whereby each line can only accommodate 16 characters. (Commas, dashes, semicolons, space, etc., are all counted as characters)





```
from machine import I2C, Pin
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd

# Initialize I2C communication
i2c = I2C(0, sda=Pin(16), scl=Pin(17), freq=400000)

# Initialize LCD display
lcd = I2cLcd(i2c, 0x27, 2, 16)

# Display "Elimu Pi" on the first line and "ROBOTICS" on the second line
lcd.putstr("Elimu Pi!\n< ROBOTICS >")
```

5) Temperature and Humidity Monitor

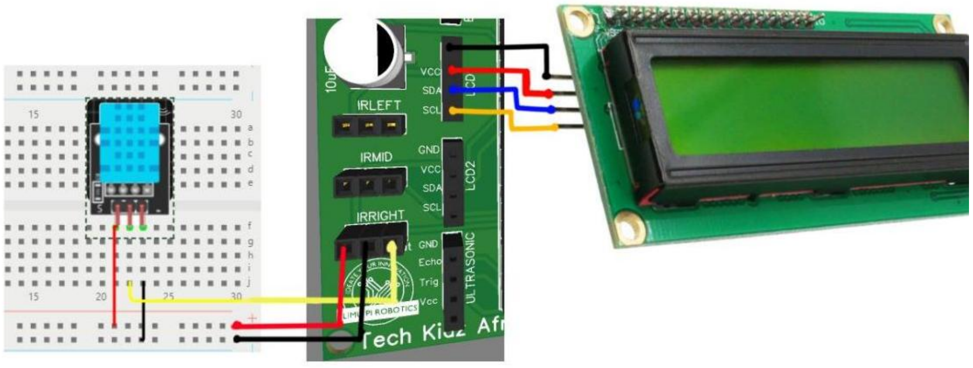
This project involves monitoring temperature and humidity using a DHT11 sensor with the Raspberry Pi Pico, and a 16x2 LCD.
To continue with this project, ensure to install the **“adafruit-circuitpython-dht”,** “**lcd_api.py**", and “**pico_i2c_lcd.py**” drivers as indicated on the **getting started** page above.

Requirements

- DHT 11 sensor
- Jumper wires.
- LCD 16x2 with i2C module.

Procedure

a) LCD 16x2 i2C module has 4 pins; GND, VCC, SDA, SCL.
b) Insert the LCD into the LCD1 connector slot.
c) On the **LCD1** slot, connect the LCD terminals as follows; **SDA ←→SDA, SCL←→SCL, 5V ←→VCC, GND ←→GND.**
d) **SCL** is connected to GPIO 3.
e) **SDA** is connected to GPIO 2.
f) The DHT11 sensor consists of 3 pins, Signal, VCC/5V, and GND.
g) Connect the VCC to any 5V pin on any connector.
h) Connect the GND to any GND on any connector.
i) Connect the signal pin to Elimu Pi’s **IR-Right OUT** pin. This OUT-pin is connected to the **GP10** of the Raspberry Pi Pico.
j) Copy the code provided to Thonny IDE and save. E.g., “***dhtsensor.py***”.
k) Click the run button.



Operation

The project uses a DHT11 sensor to obtain Humidity and Temperature values from the surrounding environment with the help of Raspberry Pi Pico which reads and processes the data obtained.
These values are displayed on the 16x2 LCD panel through the i2C adapter module.

```
import machine
import dht
from machine import I2C, Pin
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd
import time

# Initialize I2C communication for LCD
i2c_lcd = I2C(0, sda=Pin(16), scl=Pin(17), freq=400000)
lcd = I2cLcd(i2c_lcd, 0x27, 2, 16)

# Initialize DHT11 sensor
dht_sensor = dht.DHT11(machine.Pin(18))

def read_dht_values():
    dht_sensor.measure()
    return dht_sensor.temperature(), dht_sensor.humidity()

if __name__ == '__main__':
    try:
        while True:
            temperature, humidity = read_dht_values()
            lcd.clear()
            lcd.putstr("Temp: {:.1f}C".format(temperature))
            lcd.move_to(0, 1)
            lcd.putstr("Humidity: {}%".format(humidity))

            time.sleep(2)

    except KeyboardInterrupt:
        pass
```

6) Ultrasonic Distance Meter

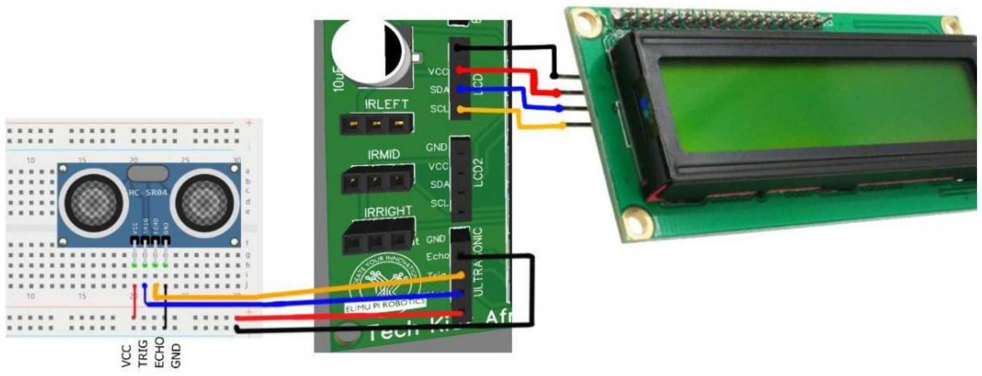
In this project, we will use an Ultrasonic sensor to measure the distance between the sensor and the object, and then display the measured distance on a 16x2 LCD.

Requirements

Ultrasonic HC-SR04
16x2 LCD with i2C module
Jumper wires

Procedure

a) Connect the LCD to the **LCD2** slot connectors.
b) For the **LCD2** slot, the **SDA** is connected to **GP6** while the **SCL** is to **GP7** on the Pico.
c) Connect the Ultrasonic sensor to the Connector slot named **Ultrasonic** on the Elimu Pi Board.
d) The **Trig** pin is connected to pin **GP15** while the **Echo** pin is connected to pin **GP14.**
e) Open Thonny IDE and install "**lcd_api.py"** and "**pico_i2c_lcd.py**" libraries to use the i2C adapter, as indicated on the **"Getting Started"** page.
f) Write the code provided below and save it e.g., "*Ultrasonic.py "* on the Raspberry Pi Pico.



Operation

With this code, the ultrasonic sensor will continuously measure distances and display the readings on the 16x2 LCD. The LCD will show the distance in centimeters.

```
import machine
import time
from machine import I2C, Pin
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd

# Initialize I2C communication for LCD
i2c_lcd = I2C(0, sda=Pin(2), scl=Pin(3), freq=400000)
lcd = I2cLcd(i2c_lcd, 0x27, 2, 16)

# Initialize ultrasonic sensor
trigger_pin = machine.Pin(15, machine.Pin.OUT)
echo_pin = machine.Pin(14, machine.Pin.IN)

def measure_distance():
    # Send a trigger pulse
    trigger_pin.low()
    time.sleep_us(2)
    trigger_pin.high()
    time.sleep_us(10)
    trigger_pin.low()

    # Measure the pulse duration on the echo pin
    pulse_duration = machine.time_pulse_us(echo_pin, machine.Pin.high(), 30000)

    # Calculate distance using the speed of sound (343 m/s)
    distance = pulse_duration * 0.0343 / 2
    return distance

while True:
    distance = measure_distance()
    lcd.clear()
    lcd.putstr("Distance: {:.2f} cm".format(distance))

    time.sleep(1)
```

7) Intruder Burglar Alarm System

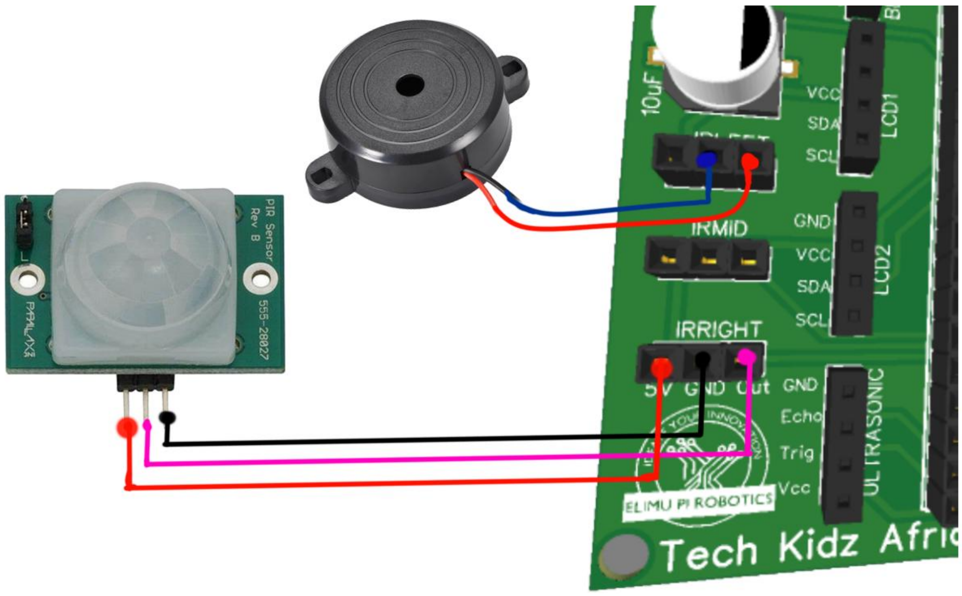
A burglar alarm detects unauthorized entry using sensors, triggering alarms or notifications for home or office security. In this project, we will use a PIR sensor and buzzer to notify the user of any intrusion.

Requirements

- PIR motion sensor
- Active buzzer
- Jumper wires

Procedure

a) For the PIR sensor, we will use the **IR_RIGHT** connecting slot. **GP12**
b) For the buzzer, we will use the **IR_LEFT** connecting slot. **GP10**
c) Connect the output pin of the PIR motion sensor to the **IR-RIGHT Out** slot, which is connected to **GP12** of the Pi Pico.
d) Connect the VCC and GND of the sensor to any of the 5V and GND pins on the Elimu Pi respectively.
e) Connect the positive (anode) pin of the buzzer to the **IR_LEFT OUT** pin slot (GP10).
f) Connect the negative (cathode) pin of the buzzer to any GND pin on the Elimu Pi board.
g) Write the code provided in Thonny IDE and save it in the Raspberry Pi as "**Burglar_alarm.py**"
h) Run the code by clicking the play/ run button on the Thonny IDE.



Operation

The PIR motion sensor detects motion and activates the buzzer to create an alarm sound. The buzzer sounds for 1 second before turning off.
When no motion is detected, the buzzer is deactivated.

```
import machine
import time

# Initialize PIR sensor and buzzer pins
pir_pin = machine.Pin(12, machine.Pin.IN)
buzzer_pin = machine.Pin(10, machine.Pin.OUT)

while True:
    if pir_pin.value() == 1:   # Motion detected
        buzzer_pin.on()
        time.sleep(0.5)   # Buzzer on for 0.5 seconds
        buzzer_pin.off()
        time.sleep(2)   # Wait for 2 seconds before checking again
    else:
        time.sleep(0.1)   # Delay between PIR sensor checks
```

8) Motion-Activated Light

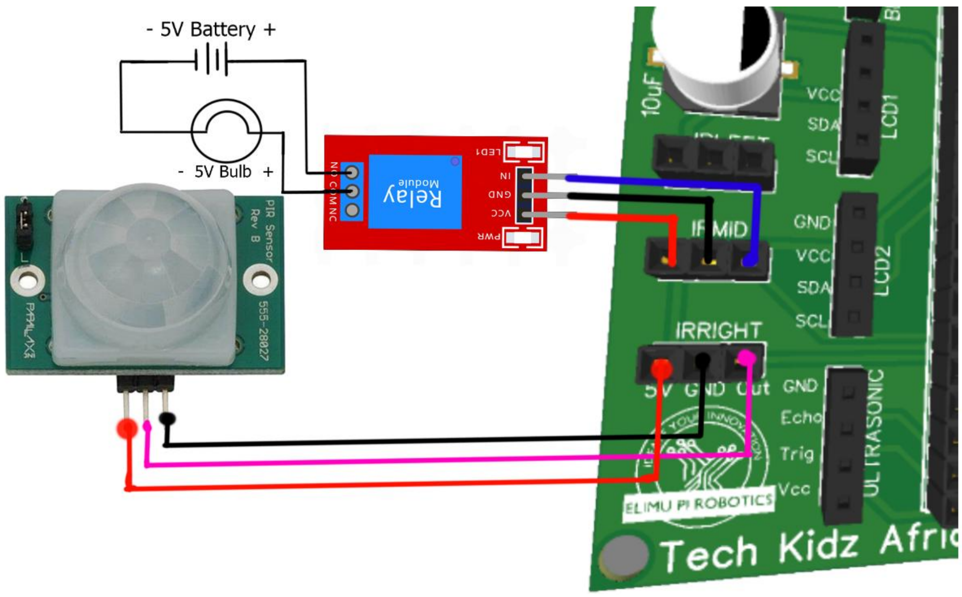
This guide demonstrates how to create motion-activated light using a PIR sensor. LED and resistor can be used as the source of light. To replace the LED with a higher voltage lamp, a relay is used to interface the Microcontroller to the lamp and power supply.

Requirements

- PIR motion sensor
- Relay module
- LED
- 220R resistor
- Jumper wires

Procedure using LED

a) For the PIR sensor, we will use the **IR_RIGHT** connecting slot. GP12
b) For the Relay/LED, we will use the **IR_MID** connecting slot. GP11
c) Connect the output pin of the PIR motion sensor to the **IR-RIGHT Out** slot, which is connected to GP12 of the Pi Pico.
d) Connect the VCC and GND of the sensor to any of the 5V and GND pins on the Elimu Pi respectively.
e) Connect one pin of the 220R resistor to the **IR_MID Out** slot which is connected to GP11 on the Pi Pico.
f) Connect the other pin of the resistor to the positive/longer pin of the LED.
g) Connect the other pin of the LED to any GND port on the Elimu Pi board.



Procedure using Relay module

h) Follow the procedure from 'a' to 'd' above.
i) Connect the input/control pin of the relay module to the **IR_MID Out** slot which is connected to GP11 on the Pi Pico.
j) Connect the VCC and GND of the relay to any 5V and GND pins on the Elimu Pi board.
k) Connect the positive terminal of the lamp to the COM of the relay module, and the other terminal to the **negative terminal of the power supply/battery.**
l) Connect the positive terminal of the battery/power supply to the N/O (Normally Open) of the relay module.
m) Write the code provided in Thonny IDE and save it on the Raspberry Pi Pico.

Operation

When the PIR senses motion within its surroundings, it sends signals to the Microcontroller, which in turn activates the relay module. This completes the external circuit where the external power supply is connected for 5 seconds before the lamp goes off.
When no motion is detected, the lamp remains off.

```
import machine
import time

# Initialize PIR sensor and relay pins
pir_pin = machine.Pin(12, machine.Pin.IN)
relay_pin = machine.Pin(11, machine.Pin.OUT)

while True:
    if pir_pin.value() == 1:   # Motion detected
        relay_pin.on()
        time.sleep(5)   # Bulb lights on for 5 seconds
        relay_pin.off()
        time.sleep(2)   # Wait for 2 seconds before checking again
    else:
        time.sleep(0.1)   # Delay between PIR sensor checks
```

9) Automatic Lighting System

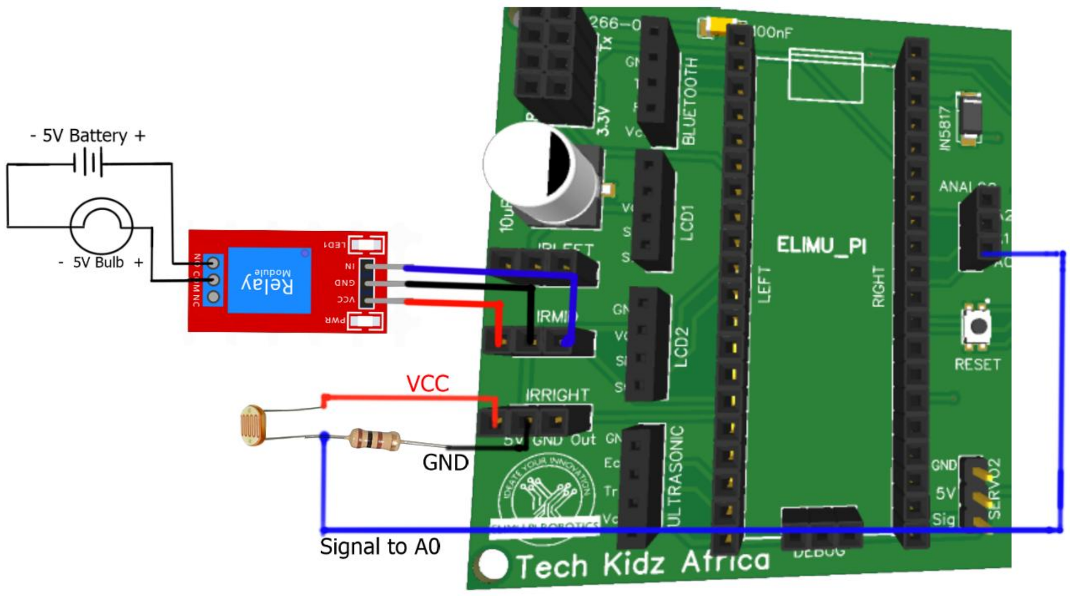
In this project, we will automate the lighting of a bulb that lights up at night and go off during the day. This is a useful remedy in reducing the cost of electricity bills.

Requirements

- LDR (Light Dependent Resistor)
- Relay module
- 5 VDC bulb
- Power supply (can be 5V)
- Jumper wires
- 1K resistor
- 10K resistor

Method

a) Connect the 1K resistor in series with the **LDR** as shown in the diagram below.
b) Use any 5V and GND ports on the Elimu Pi board.
c) Connect the **signal** pin to Analog slot **A0**. This pin is connected to **GP28** on the Raspberry Pi Pico.
d) Connect the input/control pin of the relay module to the **IR_MID Out** slot which is connected to **GP11** on the Pi Pico.
e) Connect the VCC and GND of the relay to any 5V and GND pins on the Elimu Pi board.
f) Connect the positive terminal of the lamp to the COM of the relay module, and the other terminal to the **negative terminal of the power supply/battery.**
g) Connect the positive terminal of the battery/power supply to the N/O (Normally Open) of the relay module.
h) Write the code provided below in Thonny IDE and save it on the Raspberry Pi Pico.



Operation

You might be required to change the threshold value depending on the resistor's value used in the **pull-down** connection.
When the program runs, the LDR continuously measures the light level by calculating the amount of current and resistance going to the Microcontroller. When this value is below the threshold (When there is darkness) the relay activates and the bulb goes on until when there is enough light in the surroundings (the value rises above the threshold).

```
import machine
import time

# Define pin numbers
ldr_pin = machine.ADC(machine.Pin(28))  # Analog input for LDR
relay_pin = machine.Pin(11, machine.Pin.OUT)  # Digital output for relay

# Adjust this threshold value to match your ambient light conditions
ldr_threshold = 500

while True:
    ldr_value = ldr_pin.read_u16()

    if ldr_value < ldr_threshold:
        # Turn on the relay to activate the bulb
        relay_pin.value(1)
    else:
        # Turn off the relay to deactivate the bulb
        relay_pin.value(0)

    time.sleep(1)
```

10) Robot Car Motor Control

In this project, we will control the motors' rotation using the built-in motor driver on the Elimu Pi board. Using this board, we can control the direction of rotation and speed of up to two 5VDC motors.

Requirements

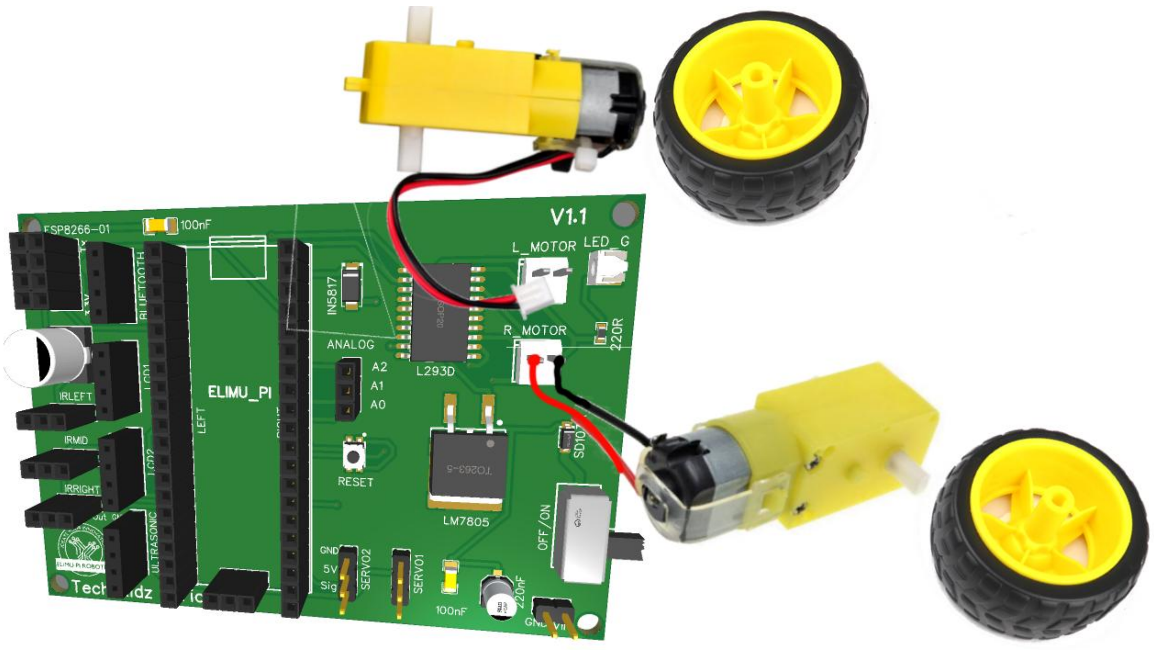
- Two 5VDC motors with motors

Method

a) On the Elimu Pi board, connect the motors on the connecting slots labeled **L_MOTOR** and **R_MOTOR**.

b) Write and save the code provided below in Thonny IDE and run it.

Left Motor			Right Motor		
Enable 1	**Input 1**	**Input 2**	**Enable 2**	**Input 3**	**Input 4**
GP8	GP18	GP19	GP22	GP20	GP21



Operation

This program demonstrates forward, backward, left, and right movements with adjustable motor speeds.
To move the robotic car forward, you need to set both the left and right motors to rotate in the same direction with the same amount of speed.
To move backward, the direction of rotation is reversed for both wheels, but keep the same speed of rotation.
To make the car move towards the right, set the left motor to forward rotation, while the right motor is set to reverse direction. Alternatively, you can set both motors to move forward, but let the rotation speed of the left motor be higher than that of the right motor.
To make the car move towards the left, set the right motor to forward rotation, while the left motor is set to reverse direction. Otherwise, you can set both motors to move forward, but let the rotation speed of the right motor be higher than that of the left motor.

```
import machine
import utime

#Define Pin outputs
Motor_Forward = machine.Pin(20, machine.Pin.OUT)
Motor_Reverse = machine.Pin(21, machine.Pin.OUT)
Motor_enable = machine.Pin(22, machine.Pin.OUT)

#Define Directions
def forward():
    Motor_Forward.on()
    Motor_Reverse.off()
    Motor_enable.on()

def reverse():
    Motor_Forward.off()
    Motor_Reverse.on()
    Motor_enable.on()

def stop():
    Motor_Forward.off()
    Motor_Reverse.off()
    Motor_enable.off()

while True:
    forward()
    utime.sleep(5)
    stop()
    utime.sleep(4)
    reverse()
    utime.sleep(5)
    stop()
    utime.sleep(4)
```

11) Home Automation Systems

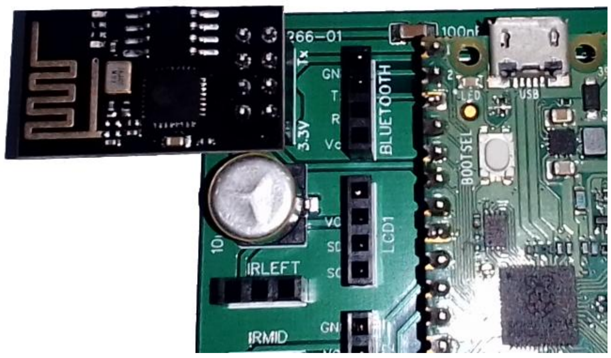
Home automation aids in monitoring and controlling various devices connected to a network remotely. This includes lighting, heating, cooling, and other appliances.

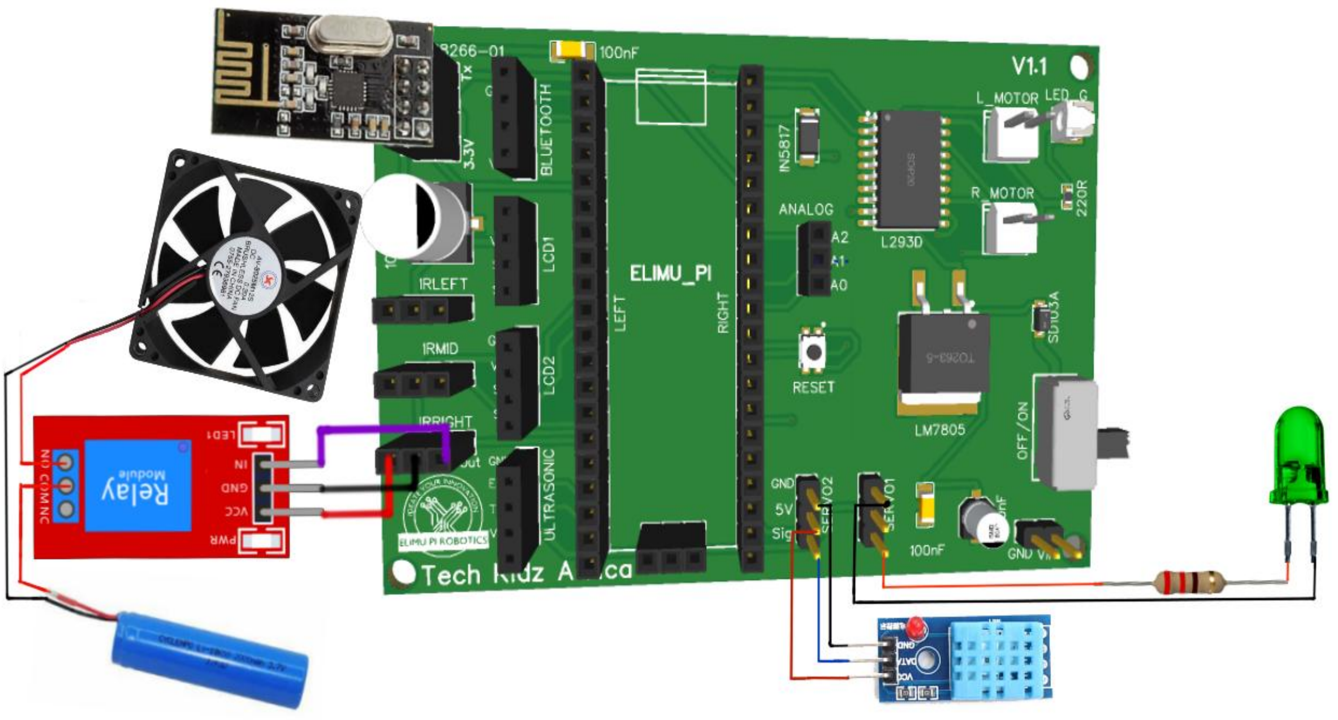
Requirements

- 5 VDC fan (motor)
- DHT11 sensor
- Relay Module
- 5V power supply
- LEDs
- 220R resistor
- ESP8266 WI-FI

Method

a) Connect the input/control pin of the relay module to the **IR_Right Out** slot which is connected to **GP12** on the Pi Pico.
b) Connect the VCC and GND of the relay to any 5V and GND pins on the Elimu Pi board.
c) Connect the positive terminal of the 5VDC motor to the COM of the relay module, and the other terminal to the **negative terminal of the power supply/battery.**
d) Connect the positive terminal of the battery/power supply to the N/O (Normally Open) of the relay module.
e) For the DHT11 sensor, connect the VCC to any 5V pin on any connector.
f) Connect the GND of the DHT11 to any GND on any connector.
g) Connect the DHT11's signal pin to Elimu Pi's **Servo2** pin. This OUT-pin is connected to **GP17** of the Raspberry Pi Pico.
h) Connect one pin of the 220R resistor to the **Servo1** slot which is connected to **GP16** on the Raspberry Pi Pico.
i) Connect the other pin of the resistor to the positive/longer pin of the LED.
j) Connect the other pin of the LED to any GND port on the Elimu Pi board.





Operation

This project enables the user to control the fan and lighting, remotely via a WI-FI connection. The Blynk Android-based app is used to send and receive commands from remote devices.

12) Weather Monitoring Station

This project involves creating a weather monitoring station for measuring and displaying various weather parameters such as temperature, humidity, and pressure.

Requirements

- HC-05 Bluetooth module
- BMP280 (Temperature and humidity)
- DHT11 sensor
- Jumper wires

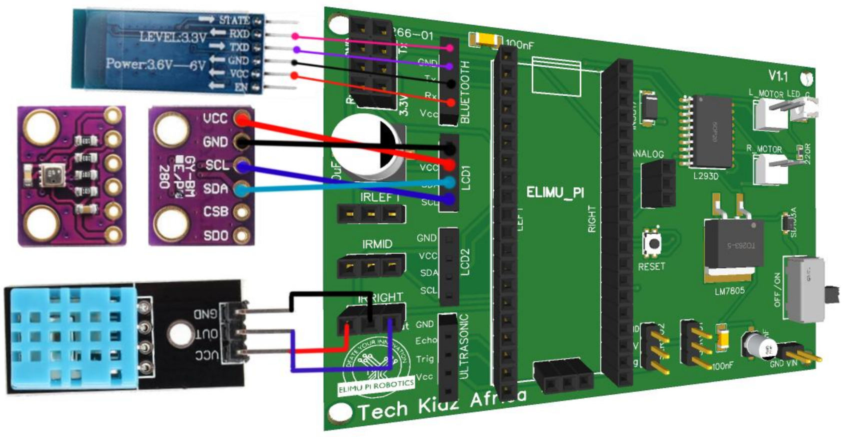
Method

a) Install the **adafruit-circuitpython-dht** library for the DHT11 sensor on your Raspberry Pi Pico.
b) Install the adafruit-circuitpython-bmp280 library for the BMP280 sensor on your Raspberry Pi Pico.
c) For the DHT11 sensor, connect the VCC to any 5V pin on any connector.
d) Connect the GND to any GND on any connector.
e) Connect the signal pin to Elimu Pi's **IR-Right OUT** pin. This OUT-pin is connected to the **GP10** of the Raspberry Pi Pico.
f) For the BMP280 sensor, connect its 5V and GND pins to any of the 5V and GND pins on the Elimu Pi Board. Connect the sensor's **SDA** and **SCL** pins to **LCD_1 SDA** and **SCL** slot.
g) For the Bluetooth module, insert it on the "**BLUETOOTH**" connecting slot provided on the Elimu Pi board. The **Tx** of the module is connected to **GP1** and **Rx** to **GP0.**
h) Pair the HC-05 module with your phone and connect.
i) Download "**Serial Bluetooth Terminal'**" from the Google Play Store and use it to receive the data being sent by the Raspberry Pi.

Bluetooth module		**BMP280 module**		**DHT11**
Tx	Rx	SCL (**LCD1**)	SDA (**LCD1**)	Signal Pin
GP1	GP0	GP3	GP2	GP10

Operation

When the program starts running, the DHT and BMP sensors obtain the temperature, humidity, and pressure values from the surrounding environment. These values are processed by the Raspberry Pi Pico which then sends them to the Bluetooth terminal app via the HC-05 module, whenever the phone and the Bluetooth module are connected.



```
import machine
import utime
import dht
import BME280
import bluetooth

# Initialize BMP280
i2c = machine.I2C(0, sda=machine.Pin(2), scl=machine.Pin(3))
bmp280 = BME280.BME280(i2c=i2c)

# Initialize DHT11
dht_sensor = dht.DHT11(machine.Pin(10))

# Initialize Bluetooth
uart = machine.UART(0, baudrate=9600, tx=machine.Pin(1), rx=machine.Pin(0))
bt = bluetooth.BLE()

def read_sensors():
    temperature = bmp280.temperature
    pressure = bmp280.pressure
    dht_sensor.measure()
    humidity = dht_sensor.humidity()
    return temperature, pressure, humidity

while True:
    temperature, pressure, humidity = read_sensors()

    # Prepare the data as a string
    data = "Temperature: {:.2f} C, Pressure: {:.2f} hPa, Humidity:
{:.2f}%".format(temperature, pressure, humidity)

    # Send data via Bluetooth
    uart.write(data + '\n')

    utime.sleep(300)  # Sleep for 5 minutes (adjust as needed)
```

13) Smart Agriculture Solutions

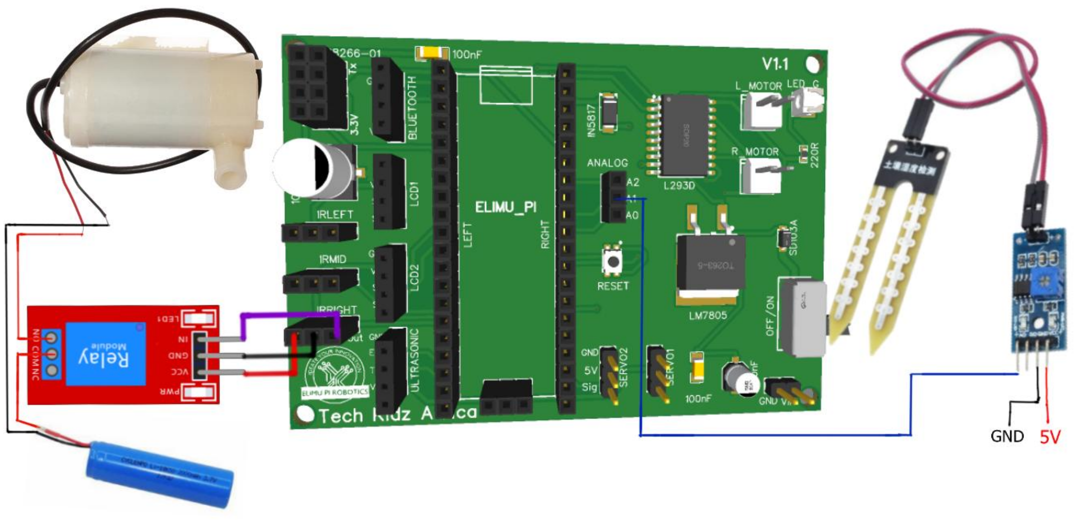
This agricultural project makes use of a soil moisture sensor to measure the amount of moisture in the soil and control the powering of the motor pump depending on the amount of moisture available.

Requirements

- 16x2 LCD with i2C module
- 5VDC relay
- 5VDCmotor pump (Micro Submersible)
- Soil moisture sensor (YL-69) and sensing probe

Method

a) The Sensor module consists of VCC, GND, D0, and A0. Connect the sensor's VCC and GND to any of the 5V and GND pins on the Elimu Pi board.
b) Connect the **A0** (Analog Pin of the sensor) to **A1** of the Elimu Pi board, which is connected to **GP27** of the Raspberry Pi Pico.
c) Insert the LCD into the **LCD1** connector slot.
d) On the **LCD1** slot, connect the LCD terminals as follows; **SDA ←→SDA, SCL←→SCL, 5V ←→VCC, GND ←→GND.**
e) **SCL** is connected to GPIO 3.
f) **SDA** is connected to GPIO 2.
i) Connect the input/control pin of the relay module to the **IR_MID Out** slot which is connected to **GP11** on the Pi Pico.
j) Connect the VCC and GND of the relay to any 5V and GND pins on the Elimu Pi board.
k) Connect the positive terminal of the DC Pump to the COM of the relay module, and the other terminal to the **negative terminal of the power supply/battery.**
g) Connect the positive terminal of the battery/power supply to the N/O (Normally Open) of the relay module.



Operation

The soil moisture sensor continuously checks the soil's moisture level.
This value is processed by the Raspberry Pi Pico through the analog pins, and if the soil becomes dry (below a set threshold), the system activates and the relay controls the water pump, providing water to the plants as the 16x2 LCD the status information.
The system runs autonomously, ensuring plants receive water only when needed, saving resources, and promoting plant health.

```
import machine
import utime
from RPi_I2C_driver import i2c_lcd

# Define GPIO pins
relay_pin = 11
moisture_sensor_pin = machine.A1

# Initialize the relay and moisture sensor
relay = machine.Pin(relay_pin, machine.Pin.OUT)
moisture_sensor = machine.ADC(moisture_sensor_pin)

# Initialize the LCD
i2c = machine.I2C(1, scl=machine.Pin(3), sda=machine.Pin(2))
lcd = i2c_lcd.i2c_lcd(i2c, 0x27, 2, 16)
lcd.backlight()

def water_plants():
    lcd.lcd_clear()
    lcd.lcd_display_string("Soil is dry.", 1)
    lcd.lcd_display_string("Watering plants...", 2)

    relay.value(1)
    utime.sleep(10)  # Run the pump for 10 seconds (adjust as needed)

    relay.value(0)
    lcd.lcd_clear()
    lcd.lcd_display_string("Watering complete.", 1)

while True:
    moisture_reading = moisture_sensor.read_u16()

    # Adjust this threshold based on your soil moisture sensor
    if moisture_reading < 20000:
        water_plants()
    else:
        lcd.lcd_clear()
        lcd.lcd_display_string("Soil is moist.", 1)

    utime.sleep(3600)  # Check moisture every hour (adjust as needed)
```

14) Smart Parking System

This project involves building a system that monitors and manages the parking spaces and displays the status of

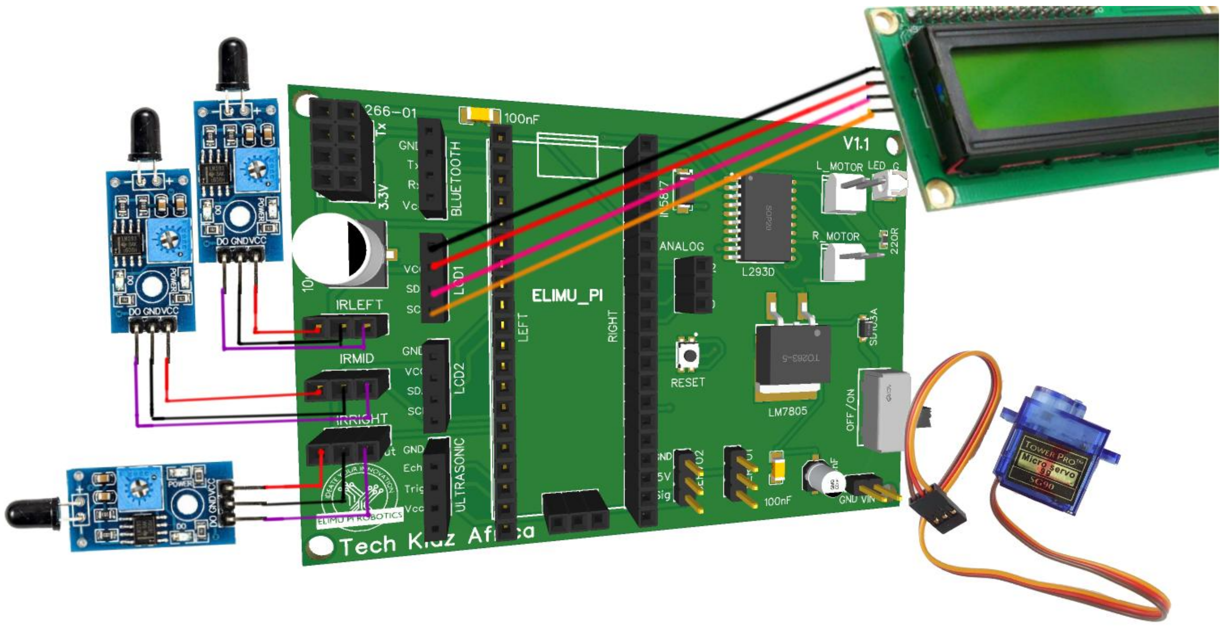
Requirements

- 3 IR sensors (HW201 IR)
- 16x2 LCD with i2C module
- Micro servo motor
- Jumper wires

Method

a) Connect the LCD to the **LCD1** slot connectors on the Elimu Pi board.
b) For the **LCD1** slot, the **SDA** pin is connected to **GP2** while the **SCL** is to **GP3** on the Raspberry Pi Pico. **SDA ←→SDA, SCL ←→SCL, 5V ←→VCC, GND←→GND.**
c) The GND and VCC/5V are connected to the power supply's GND and regulated 5Volts respectively.
d) Connect the three IR sensors to the IR slots on the Elimu Pi board, namely **IR_RIGHT**, **IR_MID,** and **IR_LEFT**.
e) Connect the servo motor to the **Servo1** slot on the Elimu Pi Board as follows; **Brown wire ←→ GND, Red wire ←→ 5V, Orange/Yellow ←→ Sig (GPIO 16.)**

IR_RIGHT	IR_MID	IR_LEFT	SDA LCD1	SCL LCD1	Servo 1 OUT
GP12	GP11	GP10	GP2	GP3	GP16



Operation

The IR sensors detect the presence of vehicles in the parking spaces. If a space is occupied, the IR sensor corresponding to that space returns a HIGH signal. Otherwise, it returns a LOW signal.
The script continuously checks the status of the parking spaces using the IR sensors. It displays the status on the LCD and opens the gate (servo motor) if at least one of the three spaces is available. If all the spaces are occupied, the gate remains closed.
The LCDs the status of each parking space and indicates whether it's "Available" or "Occupied".

```
import machine
import time
from machine import Pin, PWM
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd

# Define GPIO pins for IR sensors
ir_left = Pin(10, Pin.IN)
ir_mid = Pin(11, Pin.IN)
ir_right = Pin(12, Pin.IN)

# Define GPIO pins for servo motor
servo_pwm = PWM(Pin(16))
servo_pwm.freq(50)  # Set PWM frequency

# Define LCD settings
i2c = machine.I2C(0, sda=machine.Pin(2), scl=machine.Pin(3), freq=400000)
lcd = I2cLcd(i2c, 0x27, 2, 16)

def check_parking_spaces():
    while True:
        space1 = ir_left.value()  # 0 if empty, 1 if occupied
        space2 = ir_mid.value()
        space3 = ir_right.value()

        # Check if at least one parking space is empty
        if space1 == 0 or space2 == 0 or space3 == 0:
            servo_pwm.duty_u16(3277)  # Rotate the servo 120 degrees (out of 65535)
        else:
            servo_pwm.duty_u16(0)  # Stop the servo if all spaces are occupied

        # Display the status of parking spaces on the LCD
        lcd.clear()
        lcd.putstr("P1: " + ("Available" if space1 == 0 else "Occupied") + "\n")
        lcd.putstr("P2: " + ("Available" if space2 == 0 else "Occupied") + "\n")
        lcd.putstr("P3: " + ("Available" if space3 == 0 else "Occupied"))

        time.sleep(1)

check_parking_spaces()
```

15) Air Quality Monitoring System

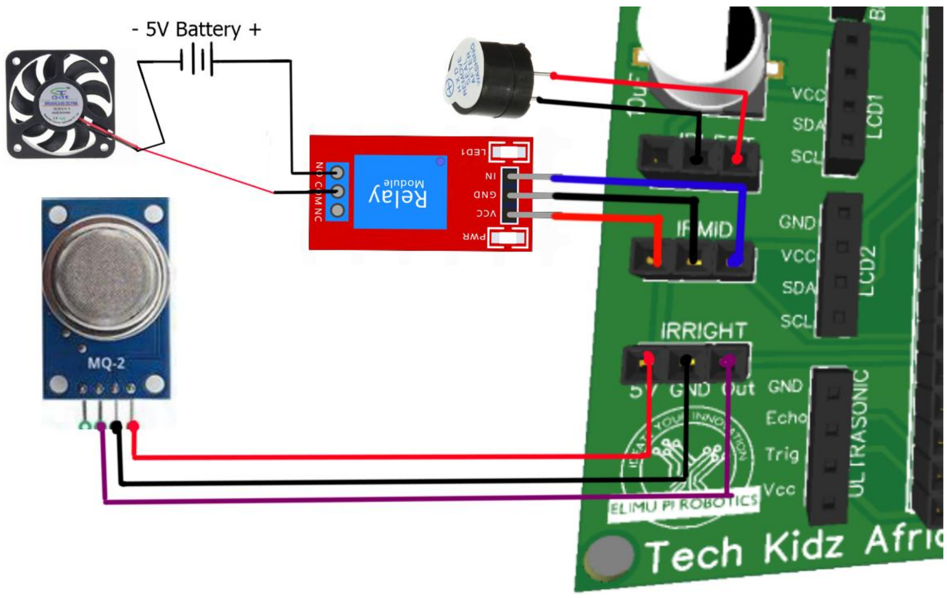
This project aims to create an air quality monitoring system that uses an MQ-2 gas sensor to detect various gases, a relay to control a 5VDC fan, and a buzzer for creating alerts.

Requirements

- 5V Relay Module
- Gas sensor MQ-2 module
- 5VDC fan
- Power supply
- Active Buzzer

Method

a) For the MQ-2 sensor, connect the pins as follows; **VCC (MQ-2) ←→ 5V (Pico), GND (MQ-2) ←→ GND (Pico)**, **DOUT (MQ-2) ←→IR_RIGHT Out** slot(**GP12**)
b) Connect the input/control pin of the relay module to the **IR_MID Out** slot which is connected to **GP11** on the Pi Pico.
c) Connect the VCC and GND of the relay to any 5V and GND pins on the Elimu Pi board.
d) Connect the positive terminal of the 5VDC motor/fan to the COM of the relay module, and the other terminal to the negative terminal of the power supply/battery.
e) Connect the positive terminal of the battery/power supply to the N/O (Normally Open) of the relay module.
f) Connect the positive (anode) pin of the buzzer to the **IR_LEFT** slot (**GP10**).
g) Connect the negative (cathode) pin of the buzzer to any GND pin on the Elimu Pi board.



Operation

The MQ-2 gas sensor measures the presence of various gases within the surrounding environment. It returns a digital signal (HIGH or LOW) based on the level of the detected gas.
The script continuously reads data from the MQ-2 sensor. If the gas level is high, it activates the relay to turn on the extractor fan and triggers the buzzer alerting the users.

```
import machine
import utime

# Define GPIO pins
MQ2_PIN = machine.Pin(12, machine.Pin.IN)      # GPIO 12
RELAY_PIN = machine.Pin(11, machine.Pin.OUT)  # GPIO 11
BUZZER_PIN = machine.Pin(10, machine.Pin.OUT) # GPIO 10

def read_gas_sensor():
    return MQ2_PIN.value()

def turn_on_fan():
    RELAY_PIN.on()

def turn_off_fan():
    RELAY_PIN.off()

def sound_buzzer():
    BUZZER_PIN.on()

def silence_buzzer():
    BUZZER_PIN.off()

while True:
    gas_detected = read_gas_sensor()

    if gas_detected:
        turn_on_fan()
        sound_buzzer()
    else:
        turn_off_fan()
        silence_buzzer()

    utime.sleep(1)
```