RASPBERRY PI 5 DEVELOPMENT KIT

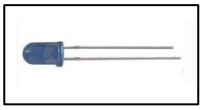
PI5-DEV-KIT

ABRA

www.abra-electronics.com

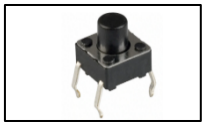
TEL: 1-800-361-5237

Parts List

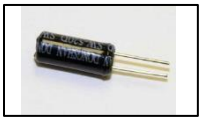


8x Blue LED

1x Passive Buzzer

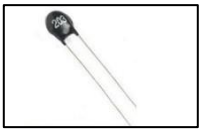


1x Push Button

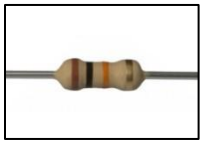


1x SW-520D Tilt Switch

1x Photoresistor

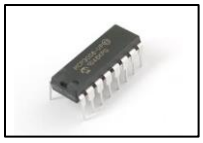


1x 10kΩ NTC Thermistor



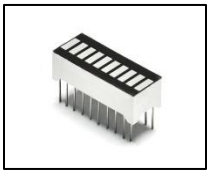
2x 10kΩ Resistor

10x 220Ω Resistor

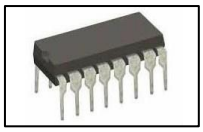


1x MCP3008 ADC IC

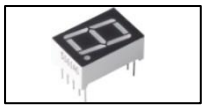
1x RGB LED



1x LED Bar Graph



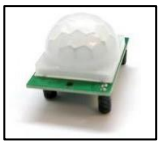
1x 74HC595



1x 7-Segment Display

1x Ultrasonic Sensor

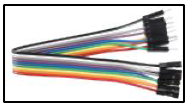
1x 4x4 Keypad



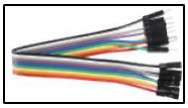
1x PIR Motion Sensor

1x 9G Micro Servo

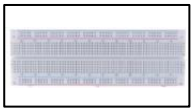
1x Motor & Driver



1x Set of 20 M-F Wires



1x Set of 20 M-M Wires



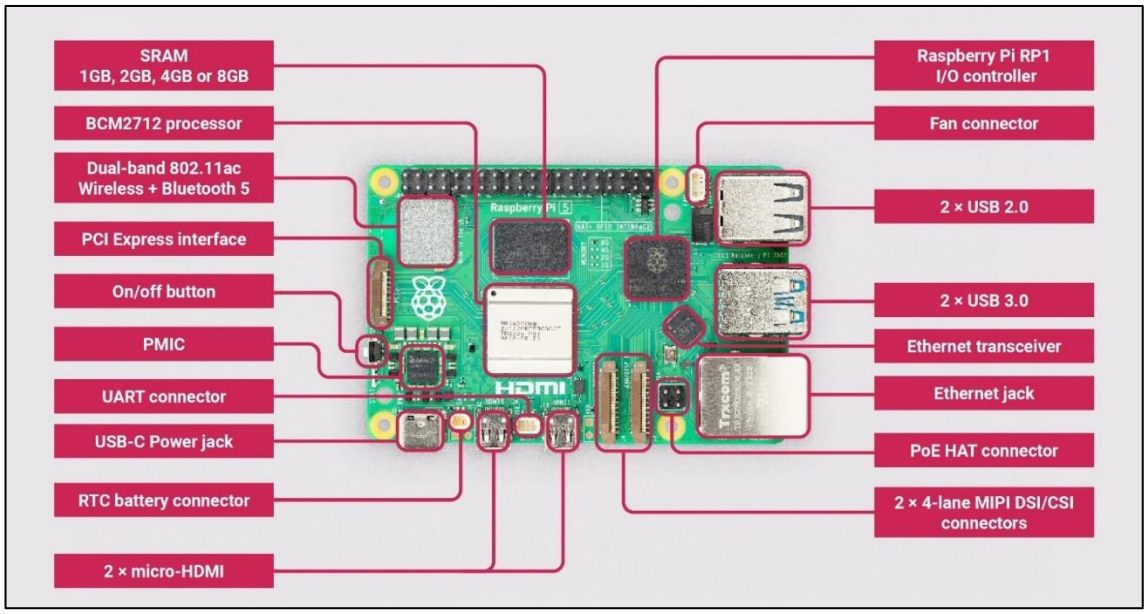
1x Breadboard

1x Resistor Card

Basic Introduction to Raspberry Pi 5

The Raspberry Pi 5 is the latest iteration of the popular Raspberry Pi single-board computer series, offering significantly improved performance and features compared to its predecessors. It is powered by a quad-core ARM Cortex-A76 CPU, enhanced GPU and faster RAM, making it suitable for a wide range of applications from simple DIY projects to more complex tasks like AI and machine learning. With upgraded connectivity options, including USB 3.0, Gigabit Ethernet, and dual-display support, the Raspberry Pi 5 continues to be a versatile and affordable choice for hobbyists, educators, and developers.

Raspberry Pi 5 Diagram



5's architecture and way of functionality vastly differs from 4B's, so note the following few key takeaways:

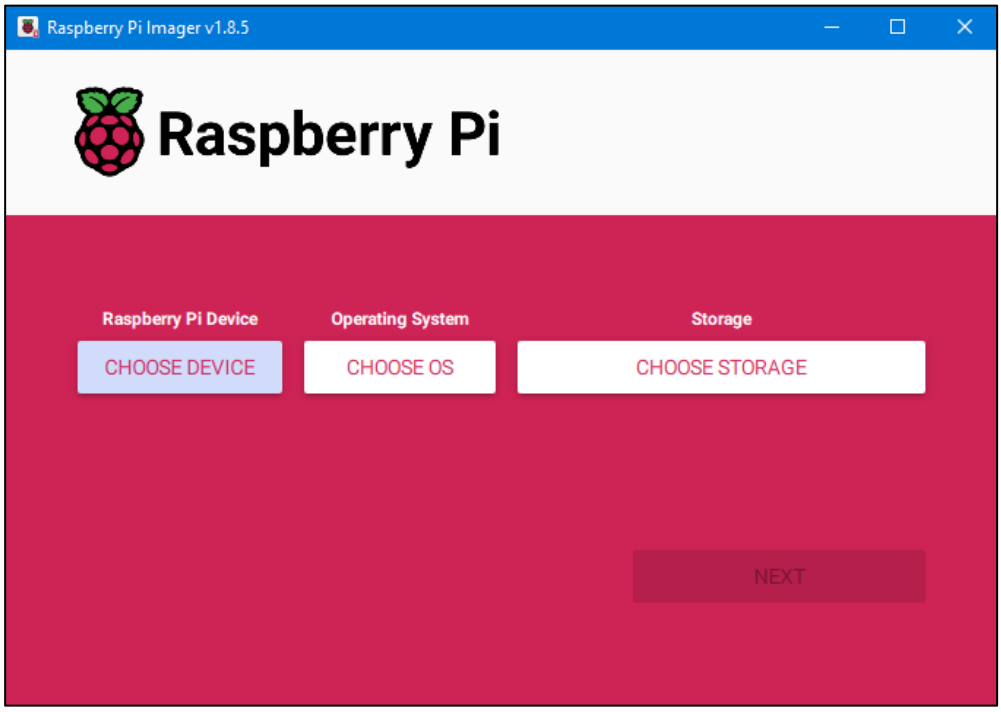
- Pi 5 requires at least 5A at 5V for a proper boot, meaning that most standard mobile phone chargers (3A at 5V) will not suffice.
- A micro-HDMI cable is now required for monitor display (micro-HDMI to HDMI).
- Pi 5 struggles to compile code using the RPi.GPIO library in Python.

Raspberry Pi 5 Operating System Setup

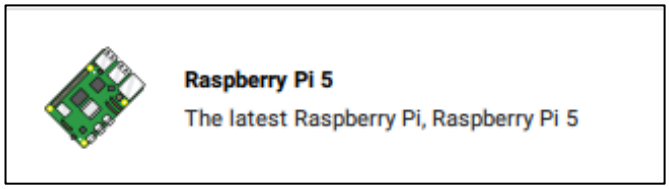
This part of the manual is dedicated to guiding you through setting up your own blank micro-SD card for Pi's use in case you do not have a pre-programmed one already. This method uses the Mini USB Micro-SD card reader. Put the card into the Mini USB and plug it into your computer.

Next, visit the following website: https://www.raspberrypi.com/software/

Download the Raspberry Pi Imager for your system and launch it.



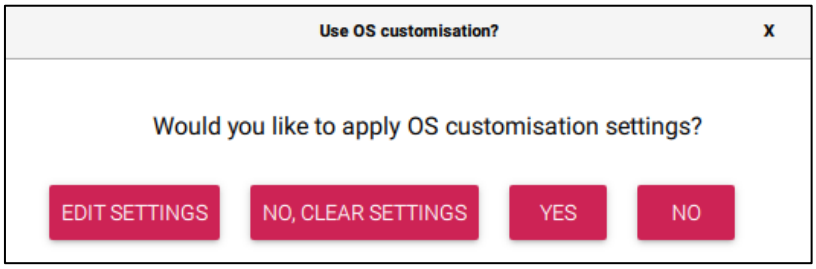
Raspberry Pi Device:



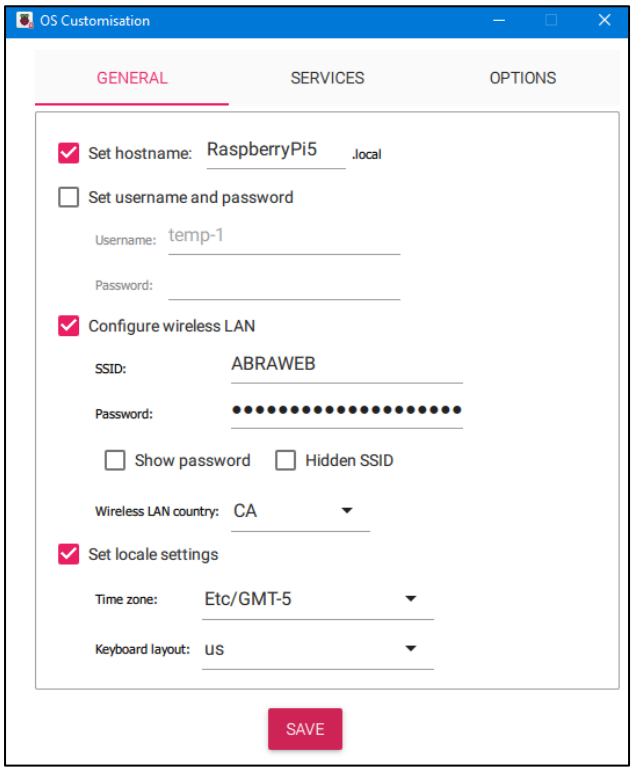
Operating System:

Storage (varies):

To minimize work on having to set up your Pi later, click EDIT SETTINGS.



In General, name your Pi device, connect it to your internet, and set up its time zone. Once finished, click Save.



Apply your OS customization settings and you will be prompted with deletion of all existing data. Click Yes and wait for writing to finish.

Once done, you may remove the micro-SD card from your computer and insert it into your Raspberry Pi.

Once you boot up your Pi with the micro-SD inserted, you will be greeted with additional setups to perform for your Raspberry Pi Desktop. Go through with its instructions until you've completed the setup and you should be prompted to restart your Pi.

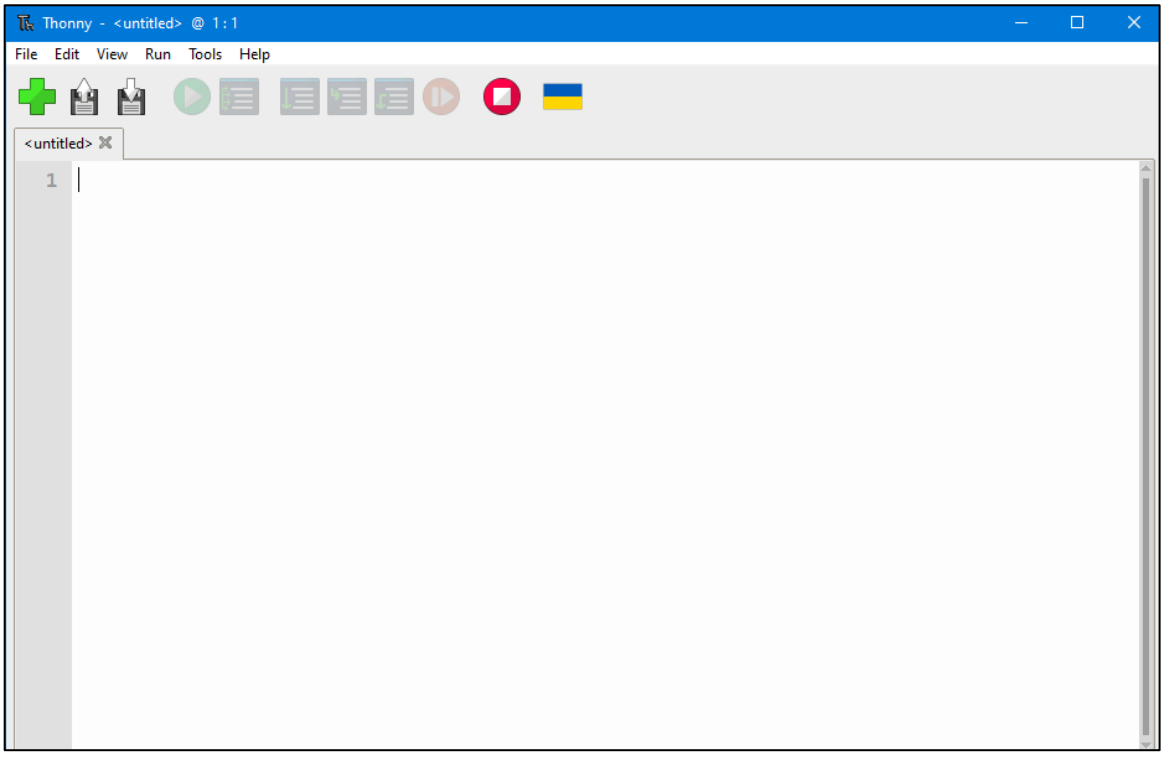
After the restart, you are officially all set up to use your Raspberry Pi 5.

Thonny

Thonny is an integrated development environment (IDE) designed for beginners and students to write Python code. It comes pre-installed with the Raspberry Pi OS and provides a simple interface for Python programming. This program will be used for running Python scripts in this kit. Simply open the .py file with Thonny.

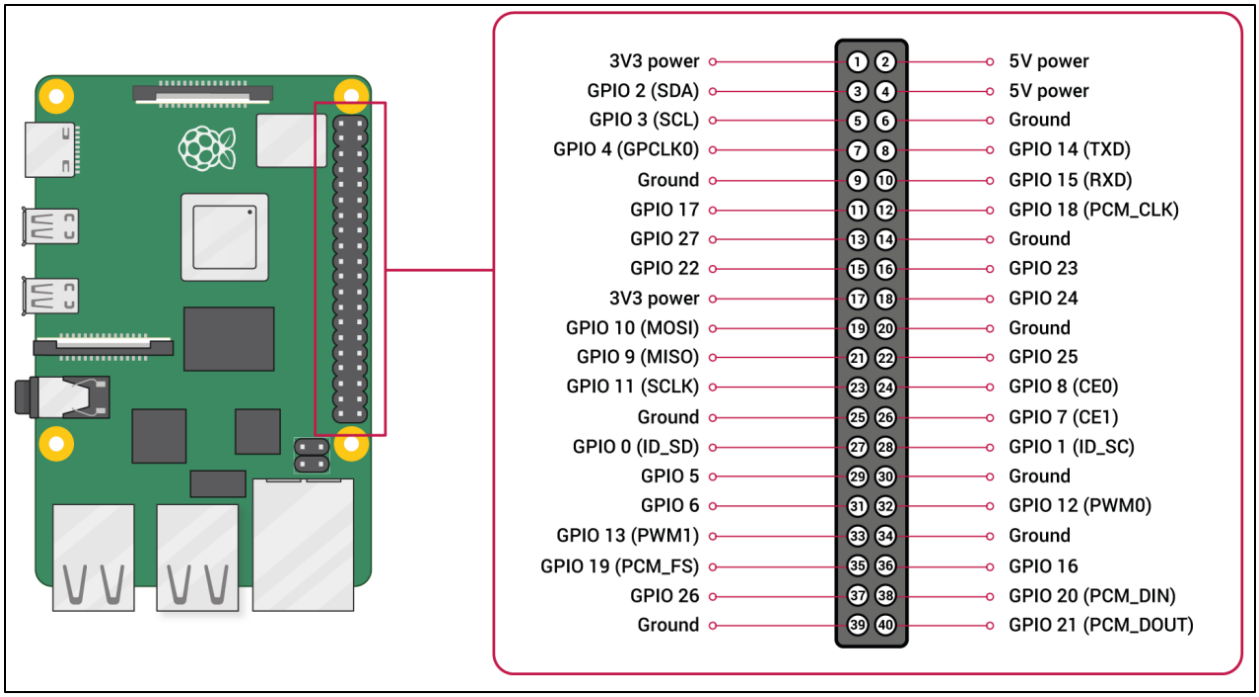
Thonny's key features:

- Simple User Interface: Ideal for beginners, with an easy-to-use interface that helps users focus on learning Python.
- Variable Value Visualization: Shows variables' values in real-time, which is helpful for debugging and understanding code.
- Step-by-Step Execution: Allows users to step through code line by line to see how Python executes it.
- Integrated Debugger: Helps in identifying and fixing errors in code.



Raspberry Pi GPIO Pinout

GPIO (General-Purpose Input/Output) pins on the Raspberry Pi 5 are physical pins on the board that can be programmed to interact with various hardware components, such as LEDs, sensors, motors, and other electronics. These pins allow the Raspberry Pi to interface with the outside world, making it a versatile tool for hardware projects.



Refer to above diagram when making connections to get an idea of what each pin is responsible for in a project.

Project 1: Blinking LED

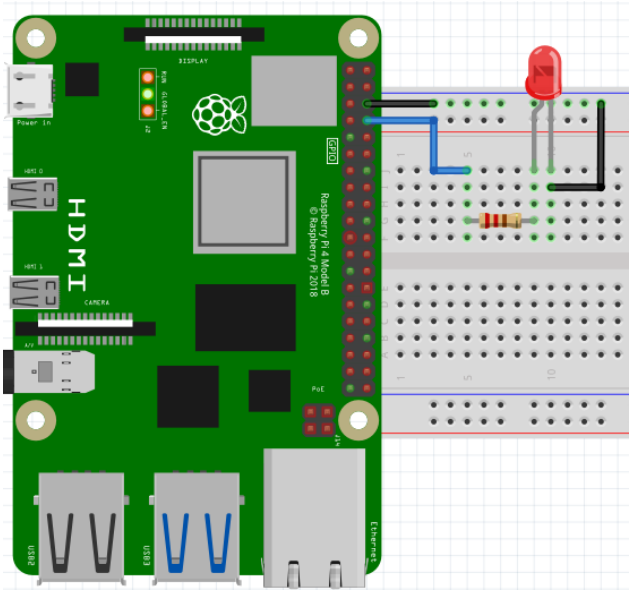
Required Components

1x Raspberry Pi 5

1x Breadboard

1x LED

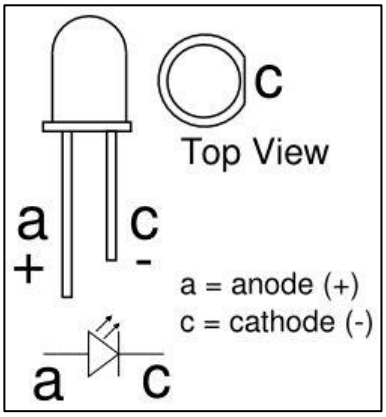
1x 220Ω Resistor



This project uses the 1-Blinking_LED.py script.

This is a classic example demonstrating how to control an LED. An LED (Light Emitting Diode) is a semiconductor device that emits light when an electric current passes through it. It has two terminals: an anode (longer pin, positive) and a cathode (shorter pin, flat spot, negative). When a voltage is applied across these terminals in the correct direction (anode positive, cathode negative), electrons flow through the LED, releasing energy in the form of light. It's very important that the LED is not placed backwards as that could break it. The responsibility of the 220Ω resistor is to limit current that passes through the LED to prevent thermal damage. Upon running the code, the LED will turn on for one second and turn off for one second, repeating indefinitely.

LED Diagram



Project 2: LED Brightness

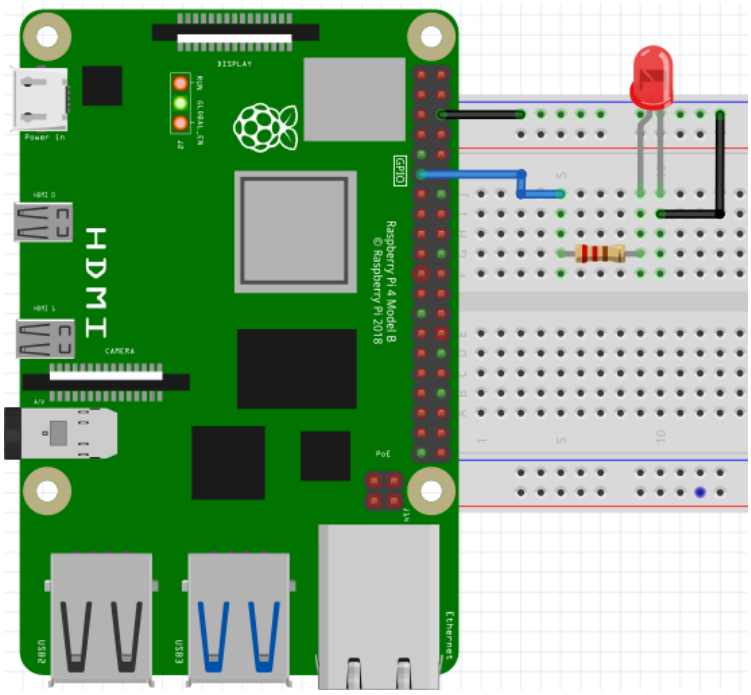
<u>Required Components</u>

1x Raspberry Pi 5

1x Breadboard

1x LED

1x 220Ω Resistor



This project uses the 2-LED_Brightness.py script.

The setup for this project is very similar to the previous at first glance, but the LED is connected to a different pin. Upon running the script, the LED's brightness increases by a certain amount every 250ms until it reaches its peak brightness. Immediately afterwards, the brightness decreases until it's off. This repeats indefinitely.

Project 3: Buzzer

<u>Required Components</u>

1x Raspberry Pi 5

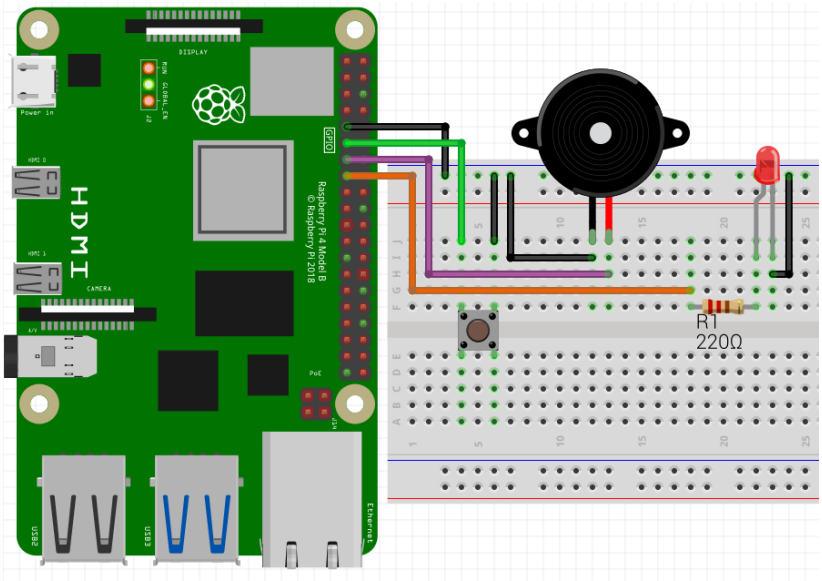
1x Breadboard

1x LED

1x 220Ω Resistor

1x Push Button

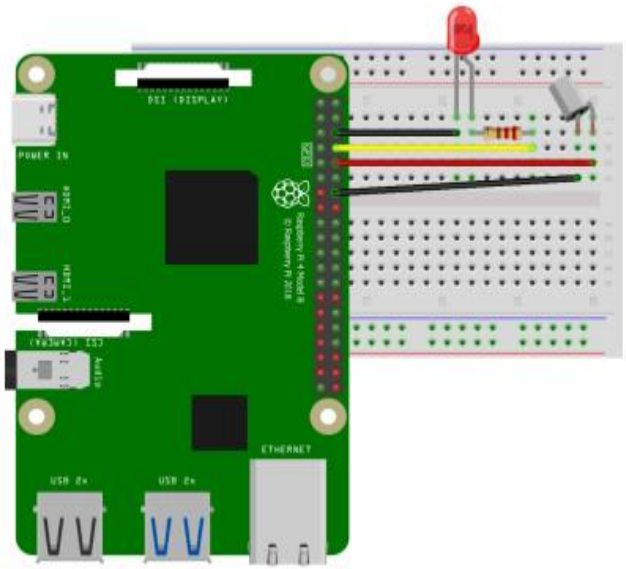
1x Buzzer



This project uses the 3-Buzzer.py script.

The active buzzer generates a tone at a fixed frequency using an internal oscillator that requires DC voltage already being provided by connected GPIO pin. Upon running the script, nothing happens until you interact with the push button. Upon pressing it, the LED will light up and the buzzer will play a tone for as long as the button is pushed. Note that the tone will not be quiet.

Project 4: Tilt Switch



<u>Required Components</u>

1x Raspberry Pi 5

1x Breadboard

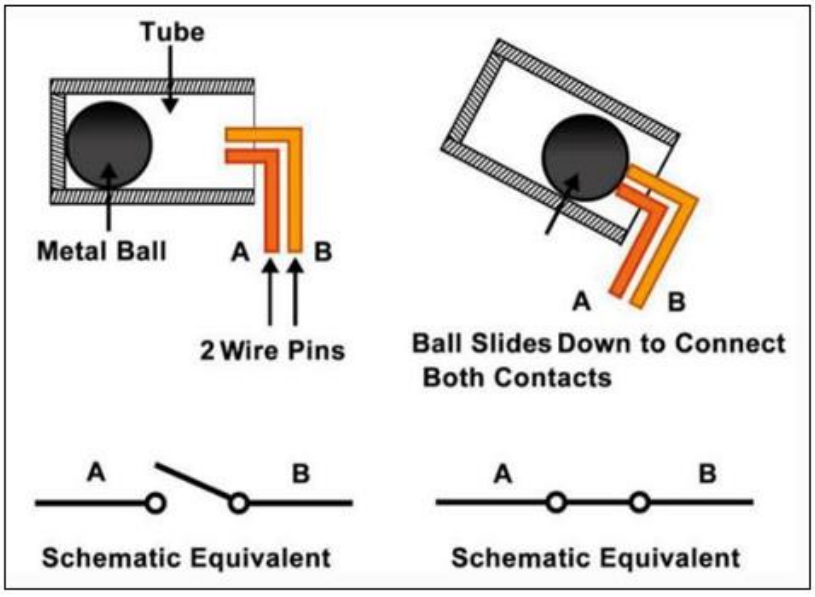
1x LED

1x 220Ω Resistor

1x Tilt Switch

This project uses the 4-Tilt_Switch.py script.

It's recommended to connect two wires directly to the tilt switch for ease of accessibility and tilting. Upon uploading the code, the light will turn on or off depending on the angle of the tilt switch. There is a little metal ball inside it that shorts the two pins. If it's upside down, it does not touch the pins and therefore will not turn the LED on.



Project 5: Photoresistor

<u>Required Components</u>

1x Raspberry Pi 5

1x Breadboard

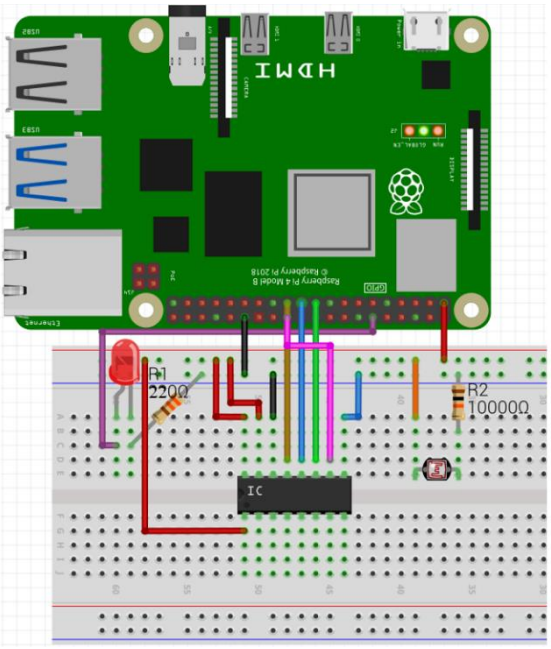
1x LED

1x 220Ω Resistor

1x Photoresistor

1x 10kΩ Resistor

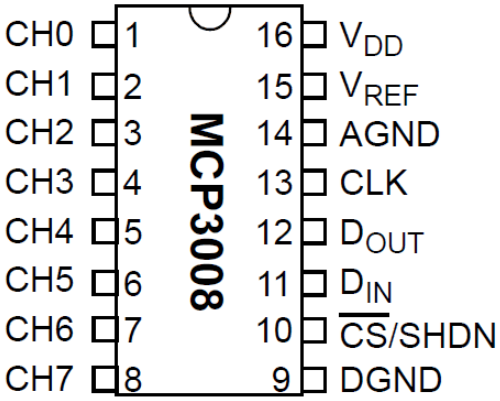
1x MCP3008 IC ADC Converter



This project uses the 5-Photoresistor.py script.

A photoresistor bears the responsibility of taking input from light that is cast onto it. Since light is described in properties of waves, Raspberry Pi cannot properly transcribe data from the photoresistor due to its lack of an onboard ADC converter. The implementation of an MCP3008 IC solves this condundrum. Upon uploading the code, the LED will light up if it's dark in your environment and will turn off if bright. Try putting your hand over the photoresistor!

MCP3008 Pinout Diagram:



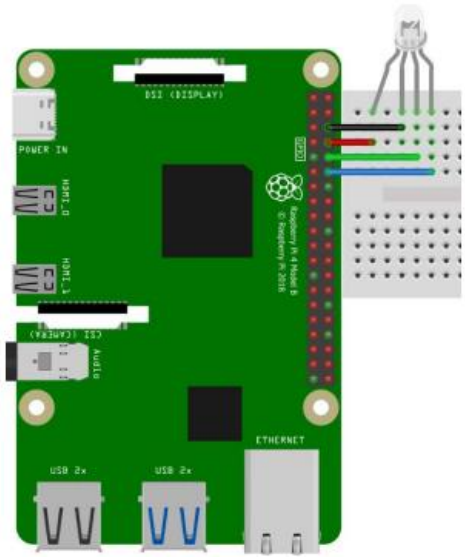
Project 6: RGB LED

<u>Required Components</u>

1x Raspberry Pi 5

1x Breadboard

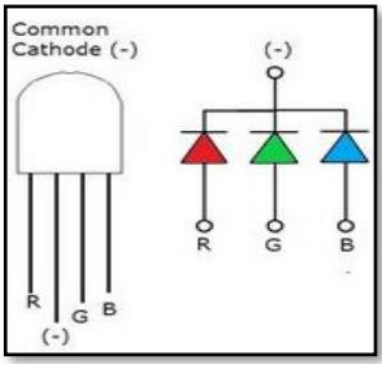
1x RGB LED



This project uses the 6-RGB_LED.py script.

A common cathode RGB LED is a type of LED that has three light-emitting diodes (red, green, and blue) within a single package. Each diode has its own anode (positive terminal), while all three diodes share a single cathode (negative terminal). To control the colors, you apply voltage to the individual anodes. By adjusting the intensity of the red, green, and blue LEDs, you can mix these primary colors to produce a wide range of colors. Since the cathode is shared, it is usually connected to the ground in a circuit. Upon pushing the code, the RGB LED will flash a different color every second in a pattern indefinitely.

RGB LED Diagram



Project 7: LED Bar Graph

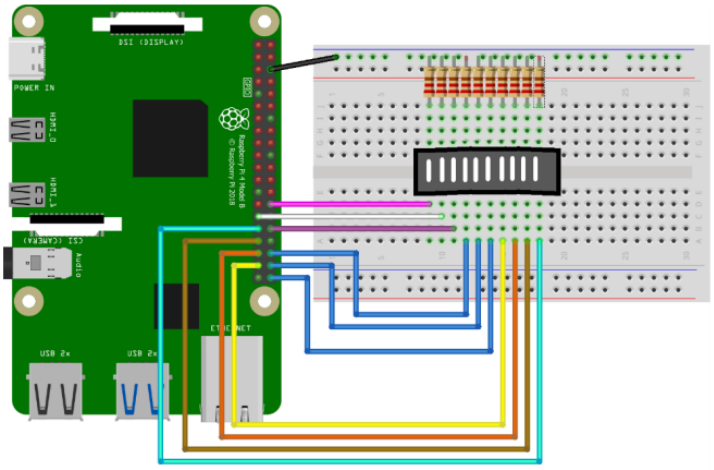
<u>Required Components</u>

1x Raspberry Pi 5

1x Breadboard

10x 220Ω Resistor

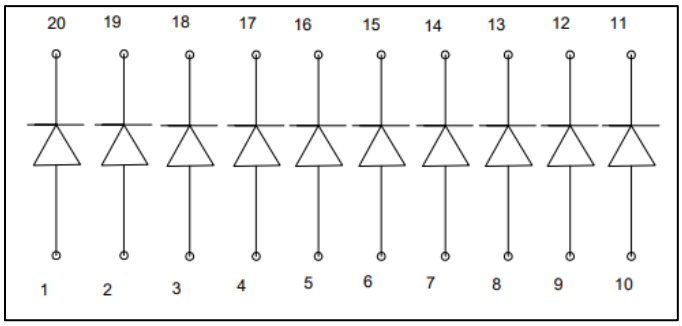
1x LED Bar Graph



This project uses the 7-LED_Bar_Graph.py script.

An LED Bar Graph is an array of LEDs typically used in electronic circuits or with microcontrollers. Connecting an LED bar graph is straightforward, with one side of the graph containing the LED anodes and the other side containing the cathodes (the anode side is labeled with the device's part number). LED bar graphs are commonly used as battery level indicators, in audio equipment, and in industrial control panels. Upon pushing the code, each LED will light up and turn off consecutively with a short delay.

LED Bar Graph's array of LEDs



Project 8: Thermistor

Required Components

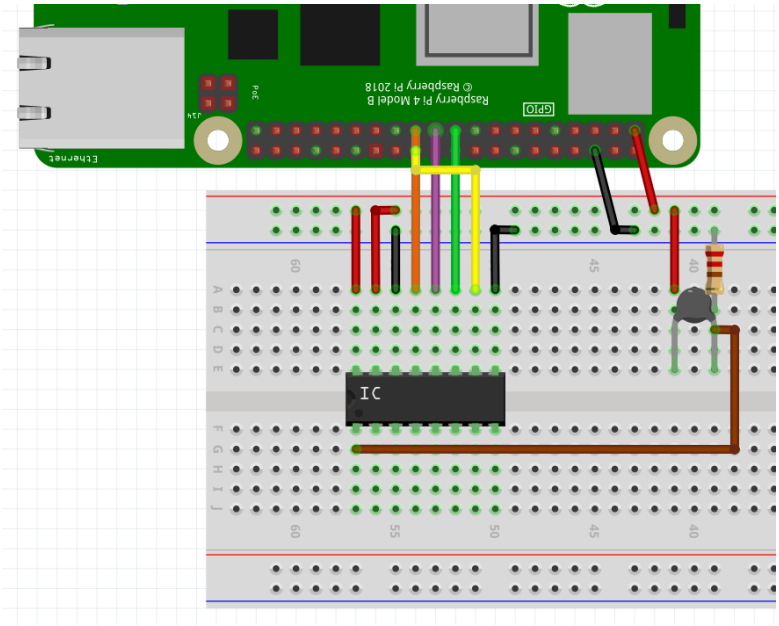
1x Raspberry Pi 5

1x Breadboard

1x 220Ω Resistor

1x Thermistor

1x MCP3008



This project uses the 8-Thermistor.py script.

A thermistor is a type of resistor whose resistance varies significantly with temperature. It is commonly used as a temperature sensor in electronic circuits. Thermistors come in two types: Negative Temperature Coefficient (NTC, in this kit), where resistance decreases as temperature increases, and Positive Temperature Coefficient (PTC), where resistance increases with temperature. Upon pushing the code, the thermistor will obtain data in similar way the photoresistor does and the MCP3008 takes care of converting that analog data into readable digital data the program performs math with, it being the temperature surrounding its environment. Read the commented lines in the code to learn how you can calculate temperature using the Steinhart equation:

$R(T)=R0 \cdot e^{(\beta(T1-T01))}$

Where:

- R(T) is the resistance at temperature T.
- R0 is the resistance at a reference temperature T0.
- β is a material-specific constant known as the Beta coefficient.
- T and T0 are in Kelvin.

Project 9: Shift Register

Required Components

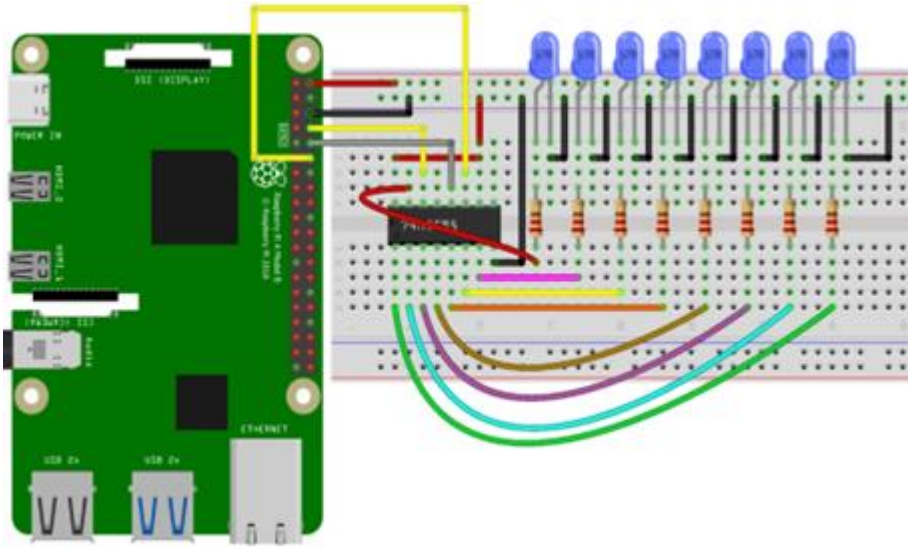
1x Raspberry Pi 5

1x Breadboard

8x 220Ω Resistor

8x LEDs

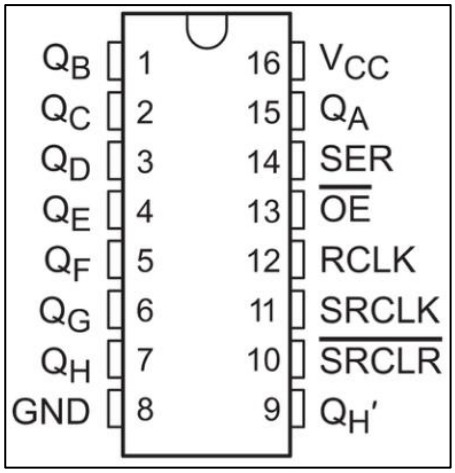
1x 74HC595



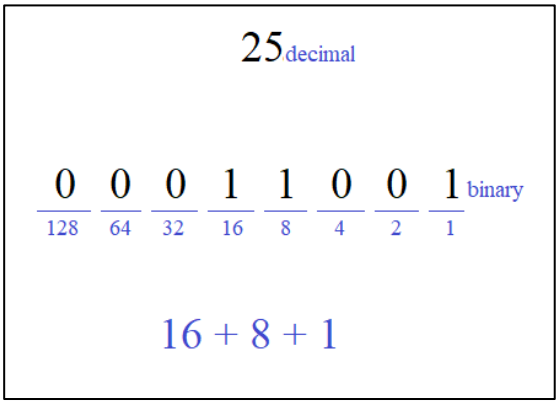
This project uses the 9-Shift_Register.py script.

The 74HC595 is an 8-bit serial-in, parallel-out shift register with a storage register and 3-state outputs. It is commonly used in microcontroller projects to expand the number of output pins, allowing you to control multiple outputs like LEDs, displays, or other components with just a few microcontroller pins. The 74HC595 takes data in serially, shifts it out to the output pins, and can latch the data to hold the output steady. It's possible to create cascaded setups, where multiple 74HC595 chips are chained together to control even more outputs. Upon pushing the code, the LEDs will begin lighting up in a pattern that imitates counting in binary.

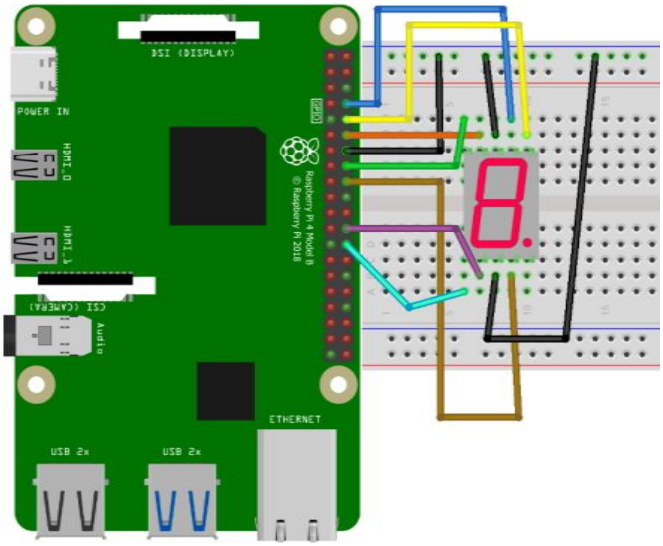
74HC595 Shift Register Pinout



Example: 25 (decimal form) expressed as 00011001 (binary form)



Project 10: 7-Segment Display



<u>Required Components</u>

1x Raspberry Pi 5

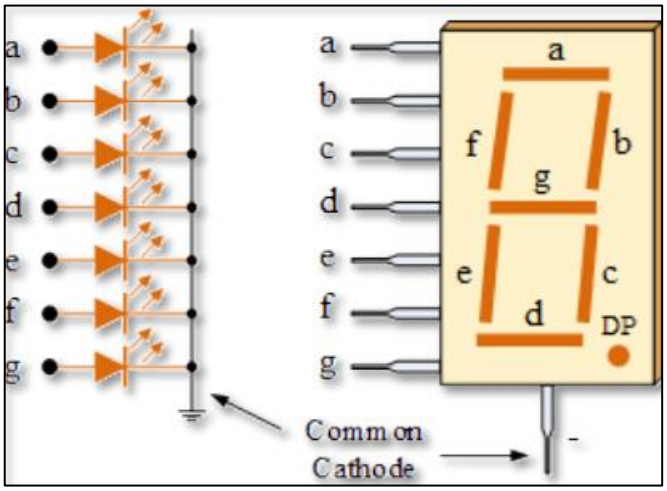
1x Breadboard

1x 7-Segment Display

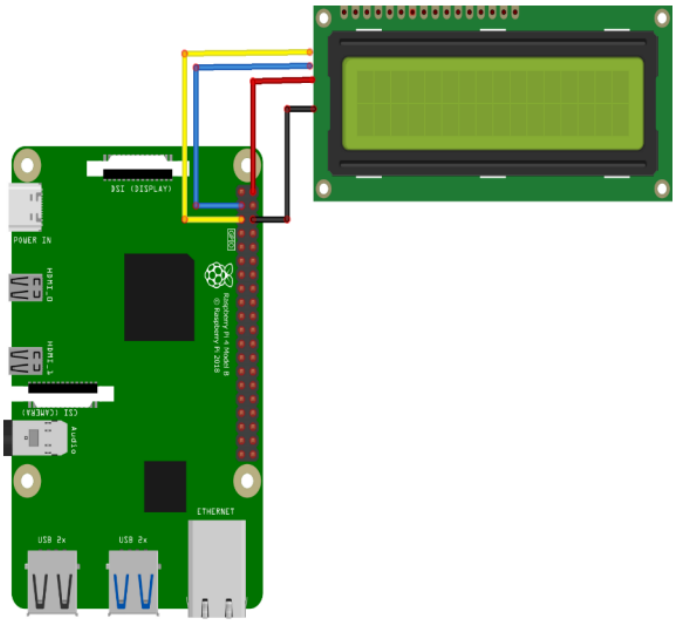
This project uses the 10-7-Segment_Display.py script.

A common cathode 7-segment display is an electronic display device used to show decimal numbers. It consists of seven LED segments arranged in the shape of the number "8" plus an additional segment for the decimal point. In a common cathode configuration, the cathodes (negative terminals) of all the LED segments are connected together to a common ground, while the anodes (positive terminals) are individually controlled. Upon pushing the code, the 7-Segment will begin counting from 0 up to F (15). The program contains a function that shows which of these segments must be enabled to display a specific number.

7-Segment Display Diagram



Project 11: I2C LCD Display



Required Components

1x Raspberry Pi 5

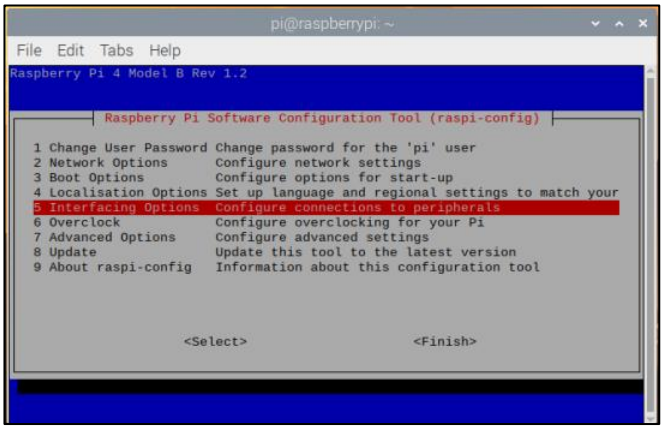
1x LCD Display

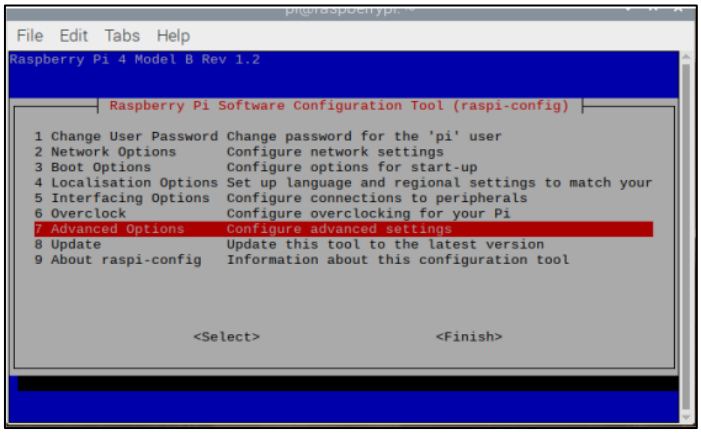
This project uses the 11-LCD_Display.py script.

An I2C 16x2 LCD display is a type of liquid crystal display (LCD) that can show 16 characters per line across two lines, making it ideal for displaying text or simple data. This display is equipped with an I2C interface, which reduces the number of GPIO pins required for control by using only two pins for communication (SDA for data and SCL for the clock).

Setting up this project requires a few more steps than the previous. First, open the console and type “sudo raspi-config”, then select Interfacing Options → Advanced Options → I2C. Select yes.

Raspberry Pi OS Configuration Tool





To run this code, an external library is required that cannot be installed on your Pi's system environment without overriding currently installed libraries, which is heavily not recommended. A simple workaround is to create and temporarily use a virtual environment for this project in the Pi's console to install the library and run the code from the console rather than Thonny.

Use the following commands:

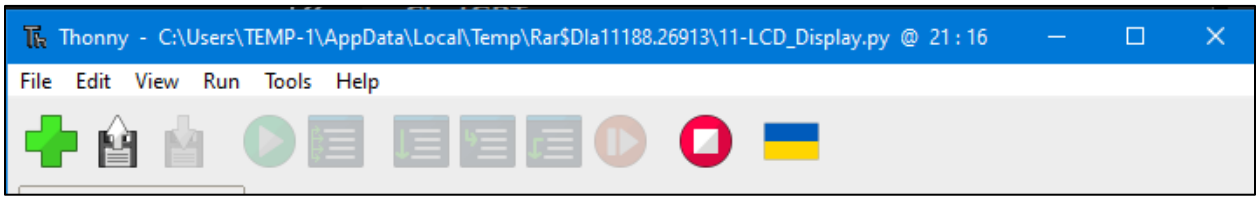
python3 -m venv myenv ← Creates the venv.

source myenv/bin/activate ← Activates the venv.

pip install RPLCD ← Installs required library.

pip install smbus2 ← Installs required module for I2C communication.

python /path/to/file/11-LCD_Display.py ← Run the project file (path can be found in Thonny's window above all the dropdown settings, will vary).



Deactivate ← Deactivates venv and returns to sys env.

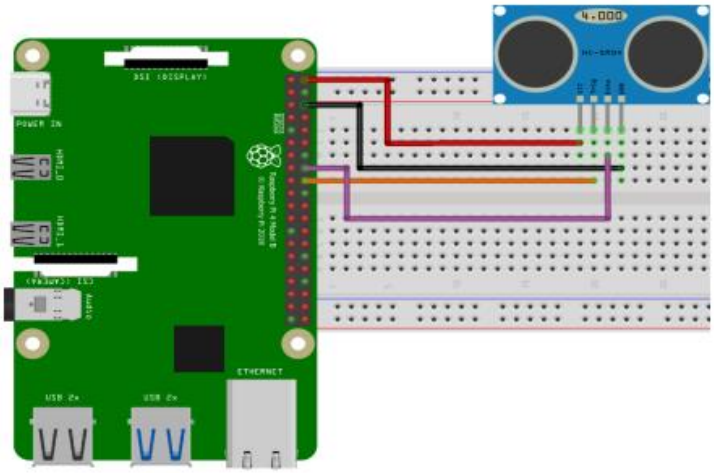
As the code is run from the console, the LCD Display will display text that was written in the programs display_text() function.

Project 12: Ultrasonic Sensor

<u>Required Components</u>

1x Raspberry Pi 5

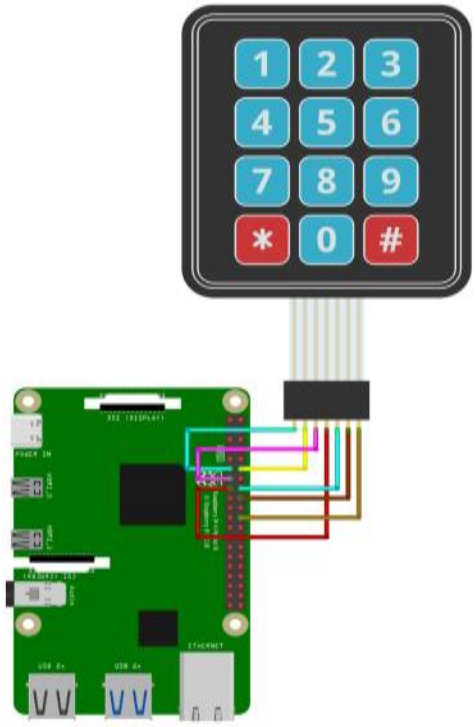
1x HC-SR04 Ultrasonic Sensor



This project uses the 12-Ultrasonic_Sensor.py script.

The HC-SR04 Ultrasonic Sensor operates by emitting an ultrasonic pulse at 40kHz and then measuring the time it takes for the pulse to bounce back after hitting an object. The sensor consists of a transmitter and a receiver. By calculating the time delay between sending and receiving the pulse, the distance to the object can be determined with a high level of accuracy (Distance = (Time * Speed of Sound) / 2). Upon pushing the code, the program will output the distance between the sensor and an object in front of it and update the output every second.

Project 13: 4x3 Keypad



<u>Required Components</u>

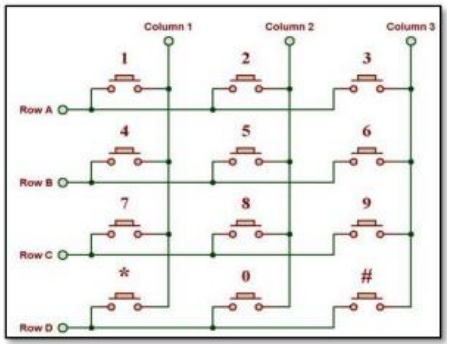
1x Raspberry Pi 5

1x 4x3 Keypad

This project uses the 13-Keypad.py script.

The 419-ADA 4x3 Keypad is a compact, membrane-style keypad featuring a 4x3 matrix layout with 12 keys arranged in four rows and three columns. Each key is labeled with numbers (0-9) and symbols (* and #). When a key is pressed, it connects a specific row to a specific column, allowing the microcontroller to detect which key was pressed by scanning the matrix. Upon uploading the code, the program will output the key that was pressed on the keypad.

Keypad's Internal Wiring Diagram



Project 14: PIR Motion Sensor

<u>Required Components</u>

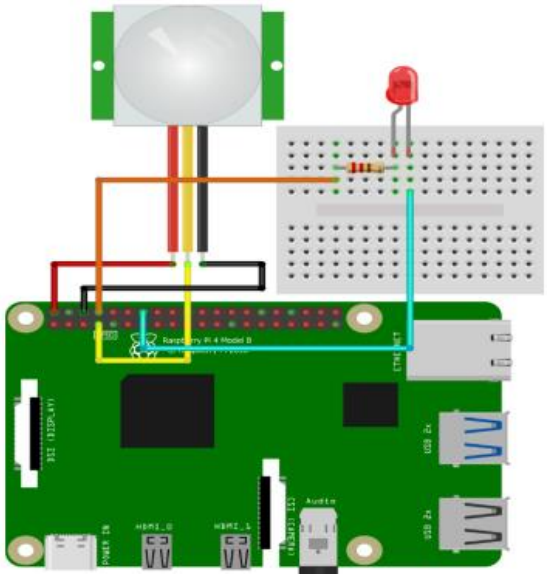
1x Raspberry Pi 5

1x Breadboard

1x 220Ω Resistor

1x LED

1x PIR Motion Sensor



This project uses the 14-PIR_Motion_Sensor.py script.

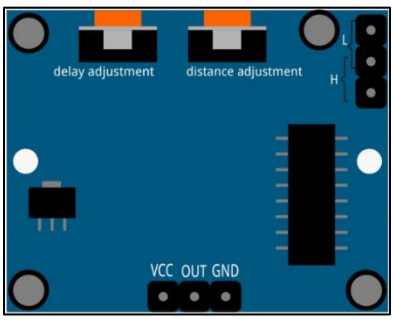
A PIR Motion Sensor works by detecting infrared (IR) radiation emitted by objects, particularly humans and animals. It contains two IR-sensitive slots that monitor the ambient IR levels. When a warm object moves in front of the sensor, it causes a difference in the IR levels detected by the two slots. This sudden change is processed by the sensor to determine if motion has occurred, triggering an output signal when movement is detected. It can be configured to meet needs:

Delay Adjustment
The sensing delay increases up to a maximum 300 seconds upon turning the knob clockwise and decreases down to a minimum of 5 seconds turning the knob counterclockwise.

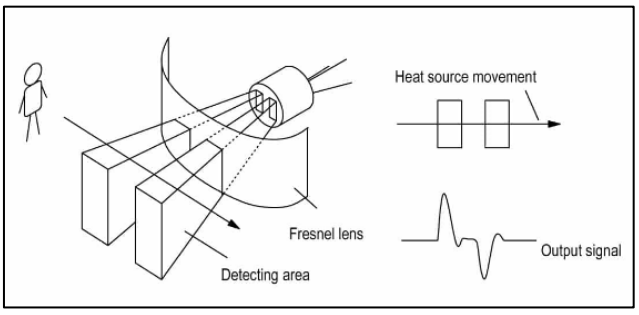
Distance Adjustment
The sensor's range increases up to a maximum of 7 meters upon turning the knob clockwise and decreases down to a minimum of 3 meters turning the knob counterclockwise.

Upon pushing the code, the program will simply output when the sensor detects motion.

Bottom View of PIR Sensor



Visualization of PIR Sensor's Functionality

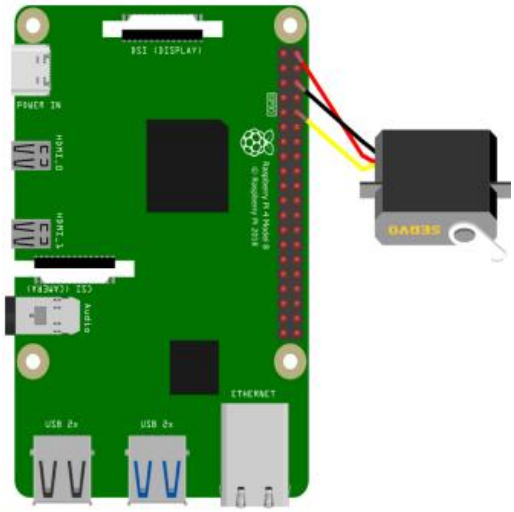


Project 15: Micro Servo

<u>Required Components</u>

1x Raspberry Pi 5

1x Micro Servo



This project uses the 15-Micro_Servo.py script.

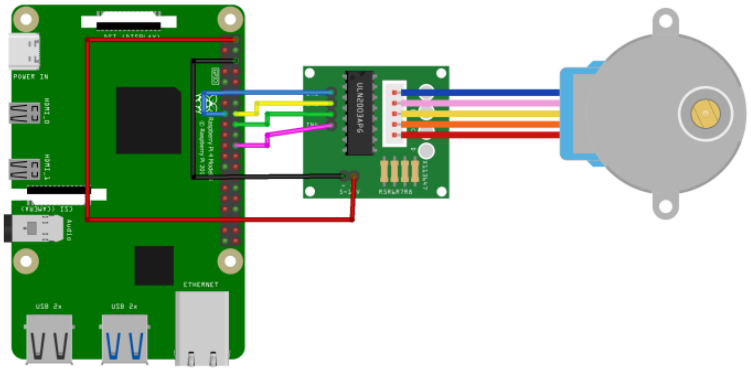
A micro servo is a small, lightweight servo motor commonly used where precise control of angular movement is needed. It typically operates on low voltage (around 4.8V to 6V) and is capable of rotating its output shaft to a specific position within a range of about 180 degrees, though some models can rotate 360 degrees. The position of the servo is controlled by a PWM (Pulse Width Modulation) signal, where the width of the pulse determines the angle of the servo's shaft. Upon pushing the code, the servo will begin its rotation in one direction, moving small steps a time, then rotate in the other direction the same way. The for-loops can be configured for desired movement of the servo.

Project 16: Stepper Motor Driver

Required Components

1x Raspberry Pi 5

1x Stepper Motor & Driver



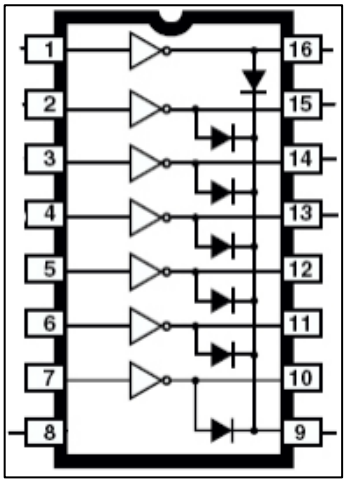
This project uses the 15-Stepper_Motor_Driver.py script.

The 28BYJ-48 is a stepper motor operating on 5V and features a 4-phase, 8-beat sequence, allowing it to rotate its shaft in precise steps. This motor has a gear reduction ratio of approximately 64:1, which gives it increased torque but slower speed, making it suitable for applications that require fine control over rotation, such as robotics, automated systems, and other mechanical movements.

To drive the 28BYJ-48 stepper motor, it is paired with a ULN2003 driver in this kit. The ULN2003 is a Darlington transistor array that provides the necessary current amplification and protection to control the stepper motor directly from a microcontroller. The driver board has four input pins connected to the microcontroller, which are used to send signals to the motor, and four output pins connected to the motor's coils. The ULN2003 board simplifies the control of the 28BYJ-48, allowing it to be driven easily by standard digital output pins from devices Raspberry Pi.

Upon pushing the code, the shaft will begin smoothly rotating clockwise and counterclockwise.

ULN2003 Motor Driver



28BYJ-48 Motor's Step Commands

8 Step : A – AB – B – BC – C – CD – D – DA
4 Step : AB – BC – CD – DA (Usual application)

Step	Hexadecimal value	IN4	IN3	IN2	IN1
A	01H	0	0	0	1
AB	03H	0	0	1	1
B	02H	0	0	1	0
BC	06H	0	1	1	0
C	04H	0	1	0	0
CD	0CH	1	1	0	0
D	08H	1	0	0	0
DA	09H	1	0	0	1