Department of Computer Science and Engineering

Course Code: CSE461	Credits: 1.5
Course Name: Introduction to Robotics Lab	Semester: Summer 23

Lab 1

Introduction to the Raspberry Pi GPIO pins, and using push buttons to control LEDs

I. Topic Overview:

The lab is designed to introduce the students to get the basic idea on the General Purpose Input-Output, or GPIO, pins of the Raspberry PI. In other words, students will be given an introduction on how the GPIO pins function and the description of each pin. Furthermore, on successfully understanding GPIO pins, they can then learn to use the GPIO pins to connect something as fundamental as push buttons. In this lab they will use push buttons to control LED lights.

III. Learning Outcome:

After this lecture, the students will be able to:

01. Know the various pins associated with the RPI
02. Program the GPIO pins to control different circuits for different usages
03. Use the GPIO pins in unison with push buttons and LEDs to create an elementary user-controlled circuit

The Raspberry Pi 4 has 40 GPIO pins that can be easily configured to read inputs or write outputs.

PIN	NAME	NAME	PIN
01	3.3V DC Power	5V DC Power	02
03	GPIO02 (SDA1,I²C)	5V DC Power	04
05	GPIO03 (SDL1,I²C)	Ground	06
07	GPIO04 (GPCLK0)	GPIO14 (TXD0, UART)	08
09	Ground	GPIO15 (RXD0, UART)	10
11	GPIO17	GPIO18(PWM0)	12
13	GPIO27	Ground	14
15	GPIO22	GPIO23	16
17	3.3V DC Power	GPIO24	18
19	GPIO10 (SP10_MOSI)	Ground	20
21	GPIO09 (SP10_MISO)	GPIO25	22
23	GPIO11 (SP10_CLK)	GPIO08 (SPI0_CE0_N)	24
25	Ground	GPIO07 (SPI0_CE1_N)	26
27	GPIO00 (SDA0, I²C)	GPIO07 (SCL0, I²C)	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12 (PWM0)	32
33	GPIO13 (PWM1)	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Here, you will be able to learn the functioning of each pin, which helps you to do things on your Raspberry Pi 4 easily. There are 40 pins in this model and among them 26 are GPIO pins.

The Raspberry Pi model includes two 5V pins, two 3.3V pins, eight ground pins and two reserved pins.

5V pins: The 5V pins are used to output the 5V power supply provided from the Type-C port. The pins are numbered 2 and 4 on Raspberry Pi 4 devices.

3.3V pins: The 3.3V pins are used to provide a 3.3V power supply to the external components which are numbered 1 and 17.

Ground pins: The ground pins are used to close the electric circuits. The ground pins help you to protect your board from burning and play an important part in a circuit. The ground pins are

numbered 6, 9, 14, 20, 25, 30, 34, and 39.

Reserved Pins: These pins are used to perform communication between I2C and EEPROM. If you are new to Raspberry Pi, you are advised not to connect anything with these pins which are 27 and 28 number pins.

GPIO Pins

These are the pins on your Raspberry Pi that perform various functions and each pin is assigned a different task. Some pins are used as inputs, while others are used as outputs. Input voltages ranging from 1.8V to 3V are considered high voltage, while voltages less than 1.8V are considered low voltage. You need to keep the voltage of the power supply below 3V in order to protect your Raspberry Pi from burning.

The GPIO pins built on Raspberry Pi devices are used to perform various functions and their details are given below

Pulse Width Modulation

The GPIO pins are used for Pulse Width Modulation (PWM), which is the process of converting a digital signal to analog signal. All pins are capable of performing software PWM, but only a few are capable of performing hardware PWM, including GPIO pins number 12, 13, 18, and 19.

Fully hardware PWM

- This type of PWM is generated by the Pi's PWM peripheral.
- The timing of the pulses is controlled by the PWM peripheral.
- It is the most accurate and arguably the most flexible.
- It can be generated on GPIO 12/13/18/19.

However there are only two channels, so only two different PWM streams can be generated at a time. GPIO 12/18 are on one channel, GPIO 13/19 on the other.
Suitable for:

- jitter free servos,
- glitch free LED brightness control,
- motor speed control.

Software Timed PWM

- This type of PWM is generated by software.
- The timing of the pulses is controlled by the (Linux) scheduler.
- The timing accuracy will vary according to the number of GPIO being used for PWM.

It is appreciably less timing accurate than fully hardware PWM or DMA timed PWM. It is much more flexible than DMA timed PWM and just as flexible as fully hardware PWM, e.g. the number of frequencies is unlimited and the number of steps between on and off is unlimited.

- Not really suitable for servos but suitable for motor control.
- Will control LED brightness but will suffer from glitches.

Serial Peripheral Interface Pins on Raspberry Pi 4

You can use Serial Peripheral Interface (SPI) pins to communicate between devices such as sensors or actuators on the Raspberry Pi. The Raspberry Pi sends data to a device via the Master Out Slave Pin (MOSI), and the same device communicates with the Raspberry Pi via the Master In Slave Out (MISO) pin. SP communication necessitates the use of five GPIO pins for GND, SCLK, MOSI, MISO, and CE. The CE pin is used to enable or disable circuit integration, whereas the SCLK pin serves as a clock for SPI communication. The Raspberry Pi's SPI communication pins are listed below.

For SPIO select GPIO9 as MISO, GPIO10 as MOSI, GPIO11 as SCLK, GPIO8 as CE0 and GPIO7 as CE1.

The SPI pins on the RPI

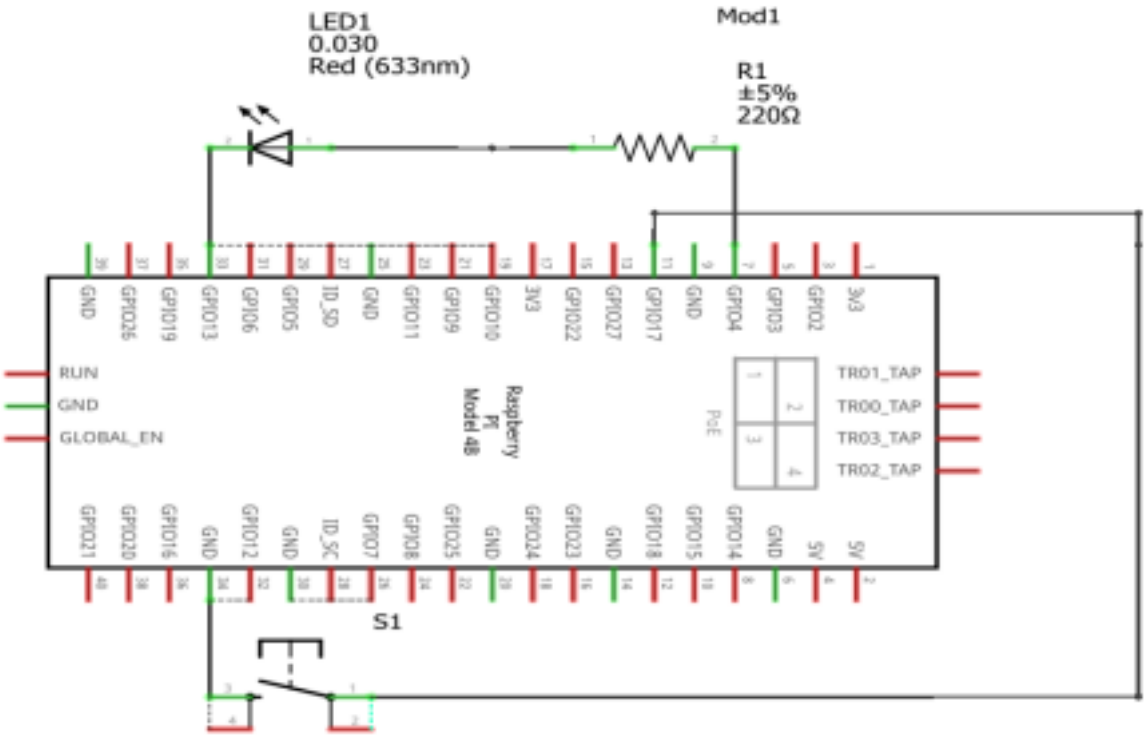
Inter Integrated Circuit (I2C) Pins on Raspberry Pi 4

Using the Inter Integrated Circuit (I2C) pins, the Raspberry Pi can easily control other peripheral devices attached with it. The communication is possible using the pins Serial Data (SDA) and Serial Clock (SCL). The data is forwarded using the SDA pin and the processing speed of data is controlled using SCL pin. There is another type of data called "Electrically erasable

programmable read-only memory (EEPROM)" data which is present in literally small quantities.

In Raspberry Pi, GPIO2 pin is responsible for transferring data using SDA and GPIO3 is used to control the speed of data by working as SCL. For the case of EEPROM, GPIO0 pin is used for data transferring while the GPIO1 pin is used as a clock to control the speed of data.

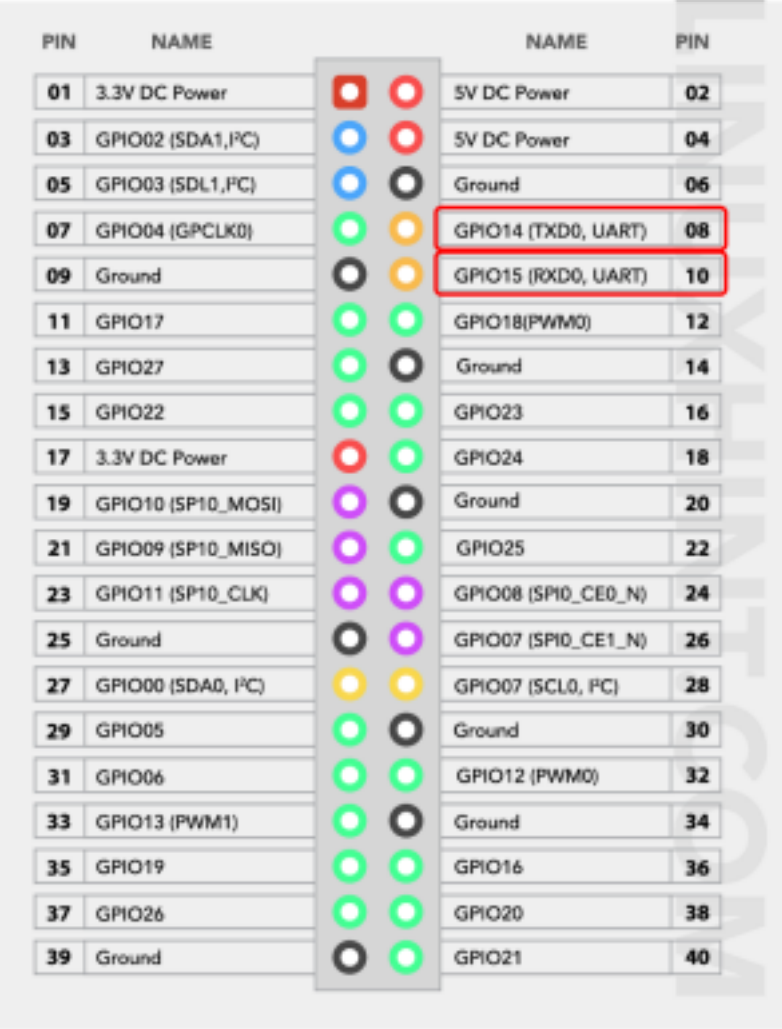
NB: Here the connection of GPIO14 will be on GND (Slight mistake correction)



The I2C pins on the RPI

UART Pins on Raspberry Pi 4

A Universal Asynchronous Receiver Transmitter (UART) is a type of communication in which data is transferred sequentially bit by bit. You need a transmitter and a receiver to perform UART. For UART communication, the Raspberry Pi 4 has two default pins. The GPIO14 pin is used as a transmitter to send data to another device, while the GPIO15 pin is used as a receiver to receive data from another device.



The UART pins on the RPI

Part II: Controlling an LED with the help of a push button

Components required for the setup:

For controlling the LED with a push button on the Raspberry Pi 4, we need the following electronic components:

- **Raspberry Pi 4**
- **LED**
- **A resistor of 220 ohms**
- **Push-button**
- **Connecting wires (female to male)**

Circuit diagram of the setup:

The code:

```
from gpiozero import LED
#imports LED functions from gpiozero library
from gpiozero import Button
#imports Button functions from gpiozero library

led = LED(4)
#declare the GPIO pin 4 for LED output and store it in led variable
button = Button(17)
#declare the GPIO pin 17 for Button output and store it in button variable

while True:
#initiated an infinite while loop
    button.wait_for_press()
#use the built-in function of the button to wait till press
    led.on()
#turn on the led
    button.wait_for_release()
#use the built-in function of button to wait till release
    led.off()
#turn off the led
```

attributes or functions in the Button Class

Name	Explanation	Parameters
`wait_for_press(timeout =None)`	Pause the script until the device is activated, or the timeout is reached.	timeout (float or None) – Number of seconds to wait before proceeding. If this is None (the default), then wait indefinitely until the device is active.
`wait_for_release(timeo ut=None)`	Pause the script until the device is deactivated, or the timeout is reached.	timeout (float or None) – Number of seconds to wait before proceeding. If this is None (the default), then wait indefinitely until the device is inactive.

`is_held`	When True, the device has been active for at least hold_time seconds.	
`is_pressed`	Returns True if the device is currently active and False otherwise. This property is usually derived from value. Unlike value, this is always a boolean.	

Lab Task

Explain the following questions:

1) Why is there a 220Ohms resistor in series with the LED?

2) Why is the push button connected from a GPIO pin on the RPI to the GND pin of the RPI instead of being connected directly to the LED and the resistor combination?

3) What would happen if the series 220Ohms resistor was replaced with a 1KOhms resistor? What visual change would you see?