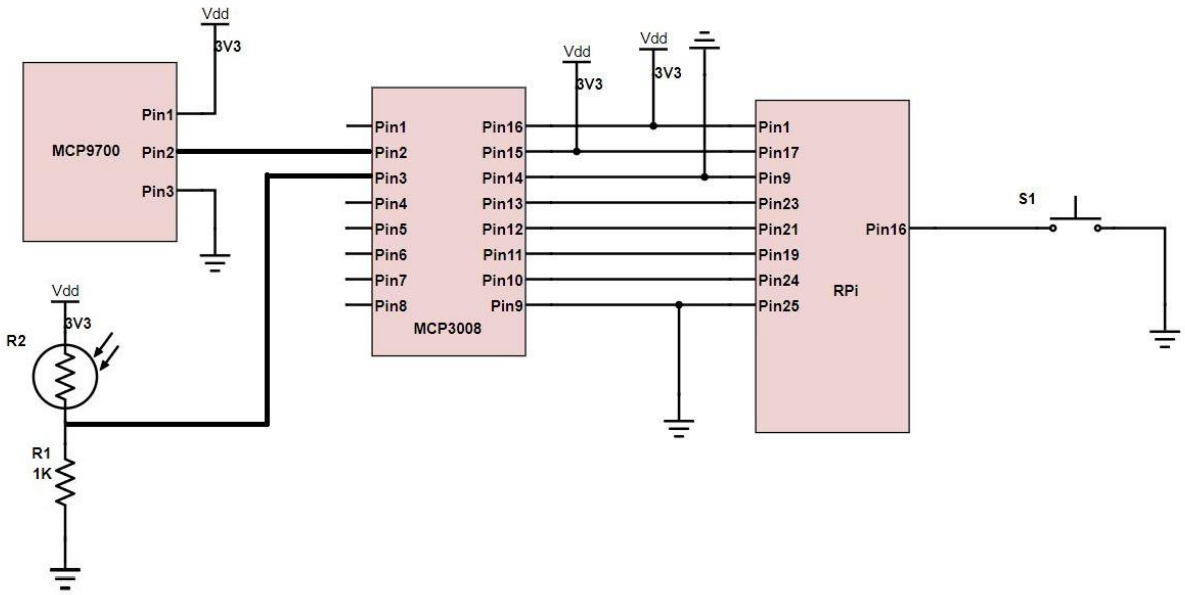
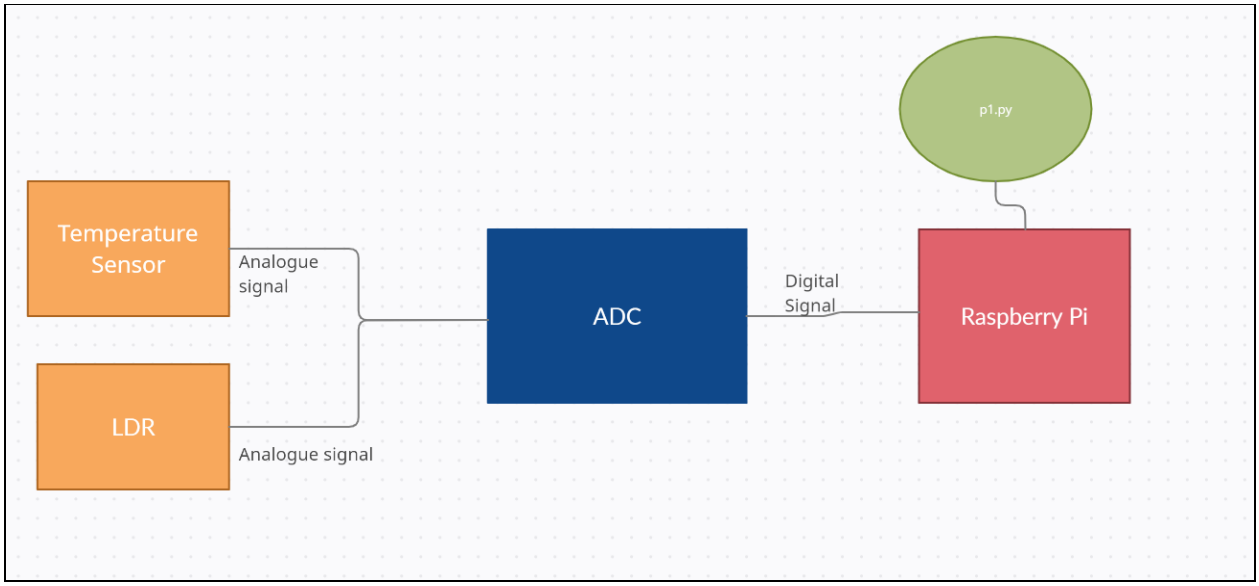
Practical 4 Report

The link to the demo video can be found here

1. The circuit diagram you used
   Below is the circuit diagram used in our implementation





2. A paragraph on your validation and testing for the values you got from the ADC for both light and temperature

The output of the ADC was validated by converting the ADC value to a temperature reading. This means each output value of the ADC corresponds to a specific temperature reading. Items with different temperatures (noticeably hot and cold) were brought close to the MCP9700 and the temperature change was observed through the output of the program. These temperature values were also validated using a thermometer that uses a similar temperature sensor to the MCP9700. The temperature readings derived from the ADC value corresponded (within the accuracy limits of the sensor) to the temperature reading on the thermometer.The output of the ADC was validated by converting the ADC value to light reading.Which also means each output value of the ADC corresponds to a specific light reading.A Flashlight with different light intensities (noticeably bright and dim) was brought close to the LDR and the light reading change was observed through the output of the program.

3. Your Python code (screenshots of code will not be marked This is the block imports different modules and initializes global variables

```
import time
import busio
import digitalio
import board
import threading
import RPi.GPIO as GPIO
import adafruit_mcp3xxx.mcp3008 as MCP
from adafruit_mcp3xxx.analog_in import AnalogIn

ldr_chan = None
temp_chan = None
sample_btn = 17
sample_rate = 1
timer = 0
```

This is the code that sets up the components used

```
def setup():

    global ldr_chan
    global temp_chan

    # create the spi bus
    spi = busio.SPI( clock = board.SCK, MISO = board.MISO, MOSI =
board.MOSI )

    # create the cs (chip select)
    cs = digitalio.DigitalInOut(board.D5)

    # create the mcp object
    mcp = MCP.MCP3008( spi, cs )

    # create analog input channels on pins 0 and 1
    ldr_chan = AnalogIn( mcp, MCP.P0)
    temp_chan = AnalogIn( mcp, MCP.P1)

    # setup sample rate button
```

```
    GPIO.setup(sample_btn, GPIO.IN)
    GPIO.setup(sample_btn, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.add_event_detect(sample_btn, GPIO.FALLING,
callback=btn_press_handler, bouncetime=400)
```

This is the code that handles the thread used for reading at a specified interval

```
def temp_ldr_thread():
    global sample_rate

    """
    Takes the readings and prints after every [sample_rate]s
    """
    thread = threading.Timer(sample_rate, temp_ldr_thread)
    thread.daemon = True
    thread.start()

    # get the current elapsed time from the timer start
    time_elapsed = (time.time() - timer)

    # get the readings from the components
    tempReading = temp_chan.value
    temp = (temp_chan.voltage - 0.5) * 100
    lightReading = ldr_chan.value

    # to_print = [runtime,"       ",tempReading,"       ",temp,"
",lightReading]
    # print(*to_print)

    # print the readings in a readable format
    print("{:<20} {:<20} {:<20}
{:<20}".format(str(round(time_elapsed)) +
"s",tempReading,str(round(temp,2)) + "C",lightReading))
```

This is the code that handles the button press event

```
def btn_press_handler(channel):
    global sample_rate
```

```
    # change the sampling rate depending on the previous rate
    if(sample_rate == 1):
        sample_rate = 5
    elif(sample_rate == 5):
        sample_rate = 10
    elif(sample_rate == 10):
        sample_rate = 1

    print("Sample rate will change to {}s".format(sample_rate))
```

Finally, this is the code that dictates what happens when the program is ran

```
if __name__ == "__main__":
    try:
        setup()
        timer = time.time()
 #start the timer
        # to_print = ["Runtime","       ","Temp Reading","
","Temp","       ","Light Reading"]
        # print(*to_print)
        print("{:<20} {:<20} {:<20} {:<20}".format("Runtime","Temp
Reading","Temp","Light Reading"))
        temp_ldr_thread()
        while True:
            pass
    except Exception as e:
        print(e)
    finally:
        GPIO.cleanup()
```

The github repo can be found here