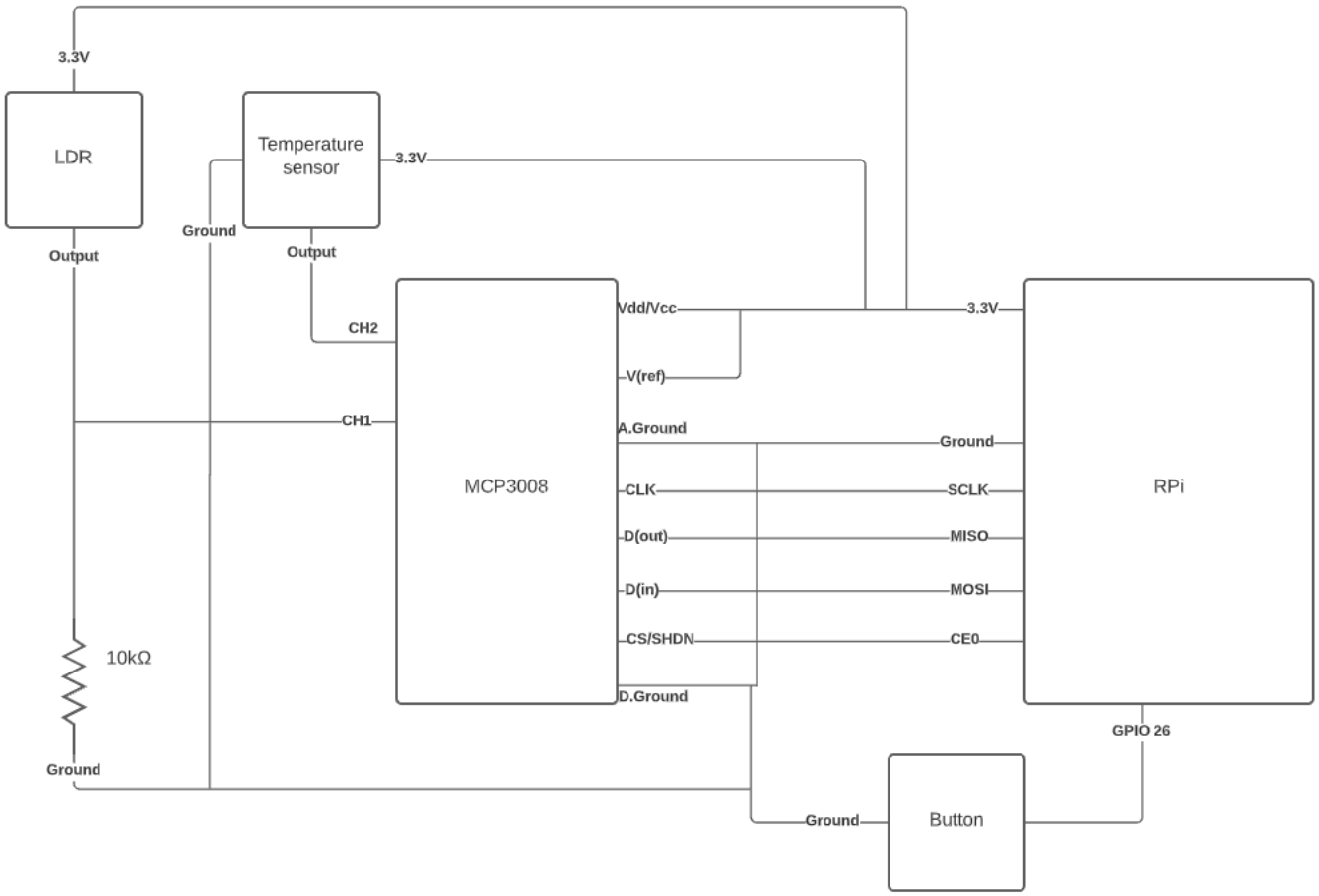
Practical 4 ADC
Done By Jonathan Campbell and Akhil Jacob

Circuit Diagram



Validation and Testing:

In the testing of the code we went step by step. First we tested the button and made sure the button was being called by using print statements. This told us where the code was as Python is a interpreted langrage and goes line by line. If there is an error with a line then a error message will appear and this will tell us where that error is.

Once the button thread is working. The print_line_threaded() can be built we used code given to us to set the MCP3008 up and we just have the threaded program printing out the values and not rerunning the MCP3008 setup as this will add more to the raspberry pi and slow it down.

To test the circuit a lighter was used to increase the temperature sensor when the time runtime intervals where at 1s. To change the LDR voltage the I used my finger to cover the sensor and block out any light and we see an immediate drop in the light reading.

Code used: (this is a text it just copies the background)

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

#Global Variables
t = 10
i = 0
time_befor = 0
Runtime = - int(time.time())

#Setup
def setup():
    print("Runtime     Temp Reading     Temp     Light Reading")
    global chan_0, chan_1, button
    # 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 an analog input channel on pin 0 and pin 1
    chan_0 = AnalogIn(mcp, MCP.P2)    #temp
    chan_1 = AnalogIn(mcp, MCP.P1)    #LDR
    #setup buttons
    button = digitalio.DigitalInOut(board.D26)
    button.direction = digitalio.Direction.INPUT
    button.pull = digitalio.Pull.UP
```

```
def btn_pressed():
    global t, i, button
    time.sleep(1)
    btn_thread = threading.Thread(target=btn_pressed)
    btn_thread.daemon = True
    btn_thread.start()
    if button.value == 0:
        # debouncing the button
        time.sleep(0.001)
        if button.value == 0:
            i += 1
            if i == 1:
                t=5.0
                pass
            elif i == 2:
                t=1.0
                pass
            elif i == 3:
                t = 10.0
                i = 0.0
                pass
        #print("button pressed  i = " + str(i) + "   t = " + str(t))
        print("button pressed")
        pass
    pass

def print_time_thread():
    global t, time_befor, Runtime, i
    thread = threading.Timer(t-0.01, print_time_thread)
    thread.daemon = True
    thread.start()
    time_now = time.time()
    Runtime += math.floor(time_now - time_befor)
    time_befor = time_now
    temp = int((chan_0.voltage-0.5)*100)
    if Runtime <= 9:
        print(str(Runtime) + "s           " + str(chan_0.value) + "          "
+ str(temp) + "C      " + str(chan_1.value))
    elif Runtime <=99:
        print(str(Runtime) + "s          " + str(chan_0.value) + "          " +
 str(temp) + "C      " + str(chan_1.value))
    elif Runtime <=999:
        print(str(Runtime) + "s         " + str(chan_0.value) + "          " +
str(temp) + "C      " + str(chan_1.value))
```

```
if __name__ == "__main__":
    try:
        setup()
        print_time_thread()
        btn_pressed()
        while True:
            pass
    except Exception as e:
        print (e)
    finally:
        GPIO.cleanup()
```

Links:

https://github.com/Jonathan5320/EEE3096S.git

(the video is in the Github folder)