Prac4_LNGANG002_BTJMAL001

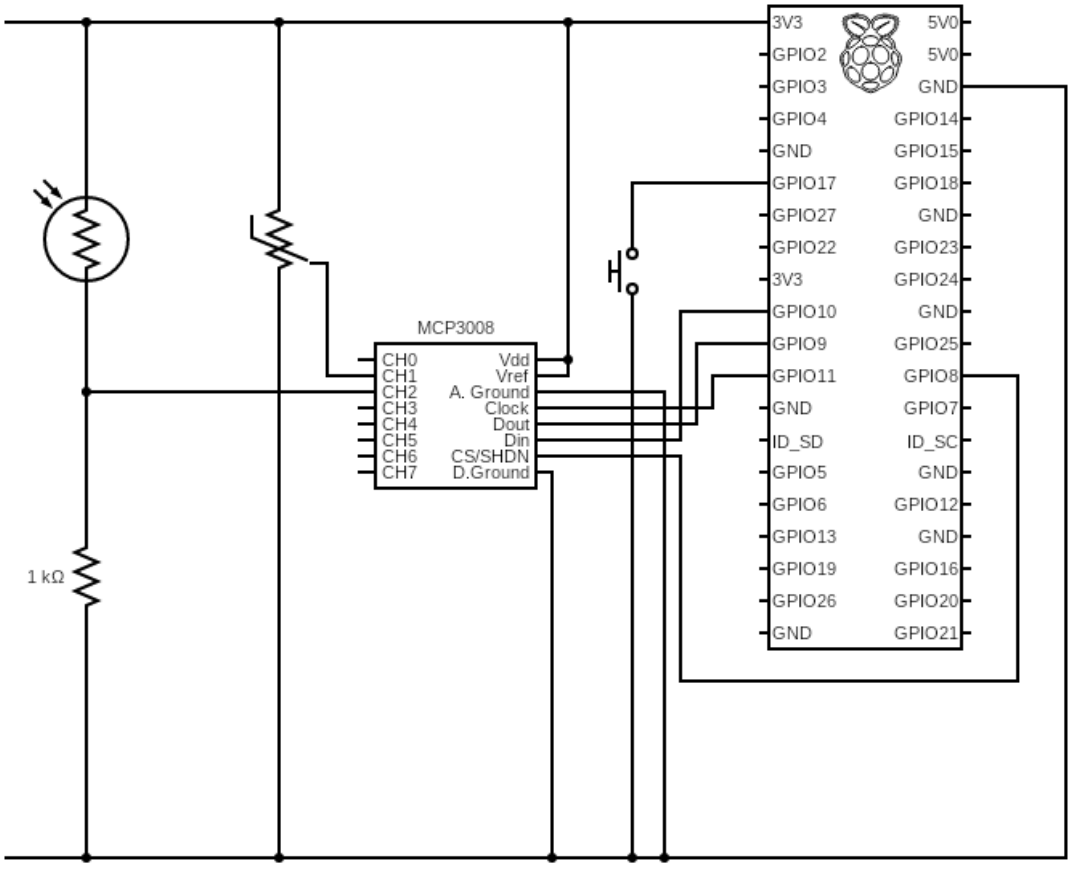
October 8, 2021

1 EEE3096S Practical 4

1.1 Date: 8 October 2021

1.2 Authors:

1.2.1 Angus Longmore (LNGANG002) and Malcom Baatjies (BTJMAL001)



1.3 Validation and Testing

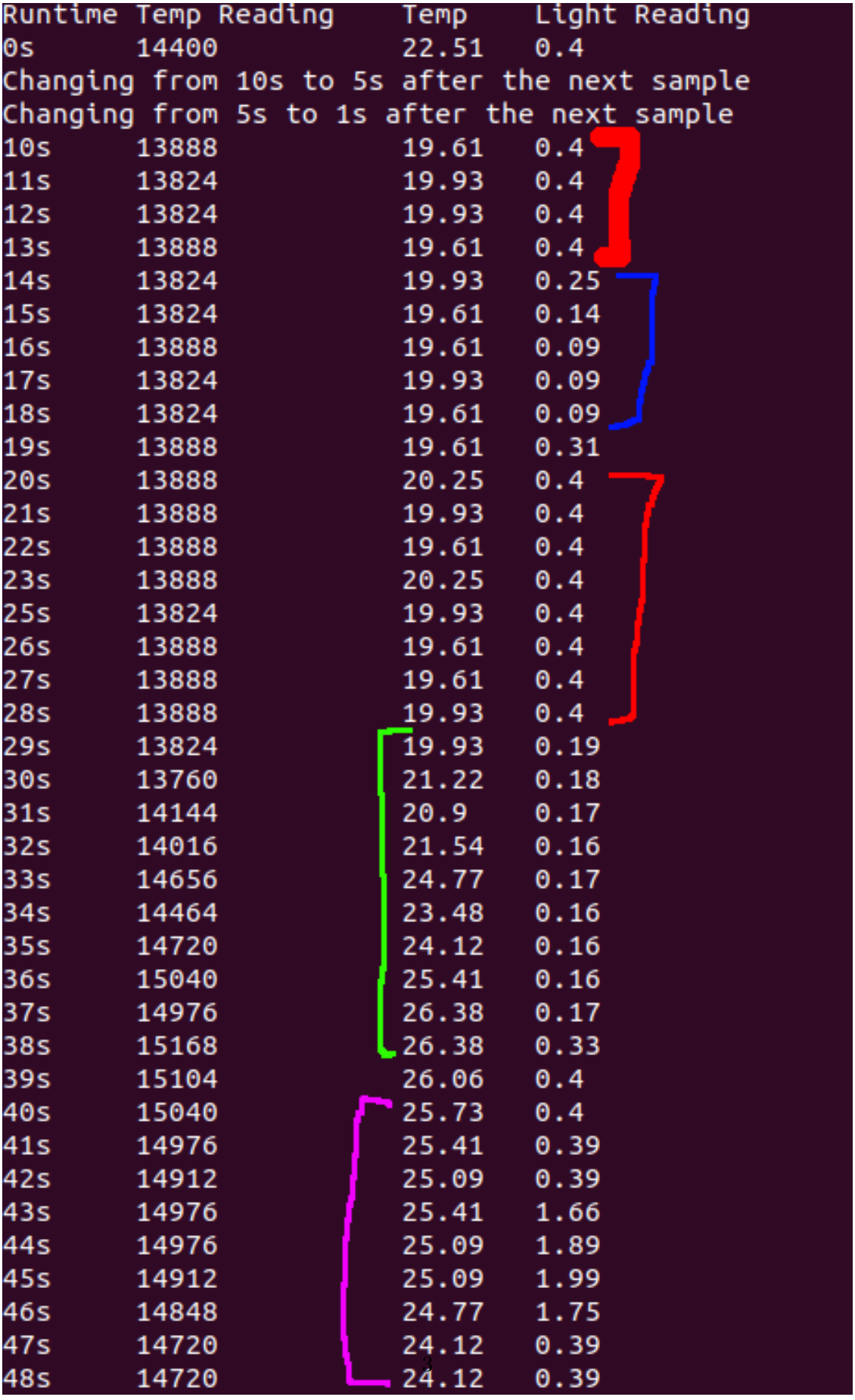
The first tests for the ldr involved measuring the readings at ambient, then covering the light, observing the readings decrease, shining a torch on the light and observing the readings increase.

To test for the Thermistor, the ADC readings were converted to Celcius, then compared to the value of a mercury thermometer. The readings were seperated by a degree and half.

The Pi was then restarted and the tests redone to validate the results. We obtained very similar readings for ambient temperature and light, the increase/decreases were also consistent with our initial testing.

The image below is a sample of some of the testing. The sections outlined in red are ambient readings.The section in blue is the light sensor being covered. The section in green is the thermistor being held by fingers (increasing the temperature), notice the ramp up as the thermistor warms. You'll also notice during this section the ldr readings drop, this is due to the ldr being close to the thermistor, and the hand of the operator blocking the light to the ldr. The pink section is when the thermistor has been released and is returning to ambient slowly.

We will demo our program in person next week, as indicated we can do on the discord server, so no video is linked.



```
Runtime Temp Reading    Temp    Light Reading
0s      14400           22.51   0.4
Changing from 10s to 5s after the next sample
Changing from 5s to 1s after the next sample
10s     13888           19.61   0.4
11s     13824           19.93   0.4
12s     13824           19.93   0.4
13s     13888           19.61   0.4
14s     13824           19.93   0.25
15s     13824           19.61   0.14
16s     13888           19.61   0.09
17s     13824           19.93   0.09
18s     13824           19.61   0.09
19s     13888           19.61   0.31
20s     13888           20.25   0.4
21s     13888           19.93   0.4
22s     13888           19.61   0.4
23s     13888           20.25   0.4
25s     13824           19.93   0.4
26s     13888           19.61   0.4
27s     13888           19.61   0.4
28s     13888           19.93   0.4
29s     13824           19.93   0.19
30s     13760           21.22   0.18
31s     14144           20.9    0.17
32s     14016           21.54   0.16
33s     14656           24.77   0.17
34s     14464           23.48   0.16
35s     14720           24.12   0.16
36s     15040           25.41   0.16
37s     14976           26.38   0.17
38s     15168           26.38   0.33
39s     15104           26.06   0.4
40s     15040           25.73   0.4
41s     14976           25.41   0.39
42s     14912           25.09   0.39
43s     14976           25.41   1.66
44s     14976           25.09   1.89
45s     14912           25.09   1.99
46s     14848           24.77   1.75
47s     14720           24.12   0.39
48s     14720           24.12   0.39
```

```
[ ]: import busio
     import digitalio
     import board
     import threading
     import RPi.GPIO as GPIO
     from time import time, sleep
     import adafruit_mcp3xxx.mcp3008 as MCP
     from adafruit_mcp3xxx.analog_in import AnalogIn

     def print_sensor_vals(start_time, temp, ldr):
         print(f"{round(time()-start_time)}s\t{temp.value}\t \t{round(((temp.
      ↪voltage-0.5)/0.01), 2)}\t{round(ldr.voltage, 2)}")


     def cycle_sample_time(channel):
         global current_sampling_time_index, sampling_times
         print(f"Changing from {sampling_times[current_sampling_time_index]}s to␣
      ↪{sampling_times[(current_sampling_time_index+1)%3]}s after the next sample")
         current_sampling_time_index = (current_sampling_time_index +1) %3

     sampling_times = [10, 5, 1]
     current_sampling_time_index = 0

     #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)

     # Add button callback to change sampling rate
     GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)
     GPIO.add_event_detect(17, GPIO.FALLING, callback=cycle_sample_time,␣
      ↪bouncetime=200)

     start_time = time()
     ldr = AnalogIn(mcp, MCP.P2)
     temp = AnalogIn(mcp, MCP.P1)
     print("Runtime\tTemp Reading\tTemp\tLight Reading")
     while True:
         x = threading.Thread(target=print_sensor_vals, args=(start_time, temp, ldr))
         x.start()
         x.join()
         wait_time = sampling_times[current_sampling_time_index]
```

```
    sleep(wait_time)
```