Lab 2: Using GPIO for Input/Output Report

Tyler Nardecchia
Yerania Hernandez
CSCE 462 - 502
September 18, 2017

Thinking and Exploring

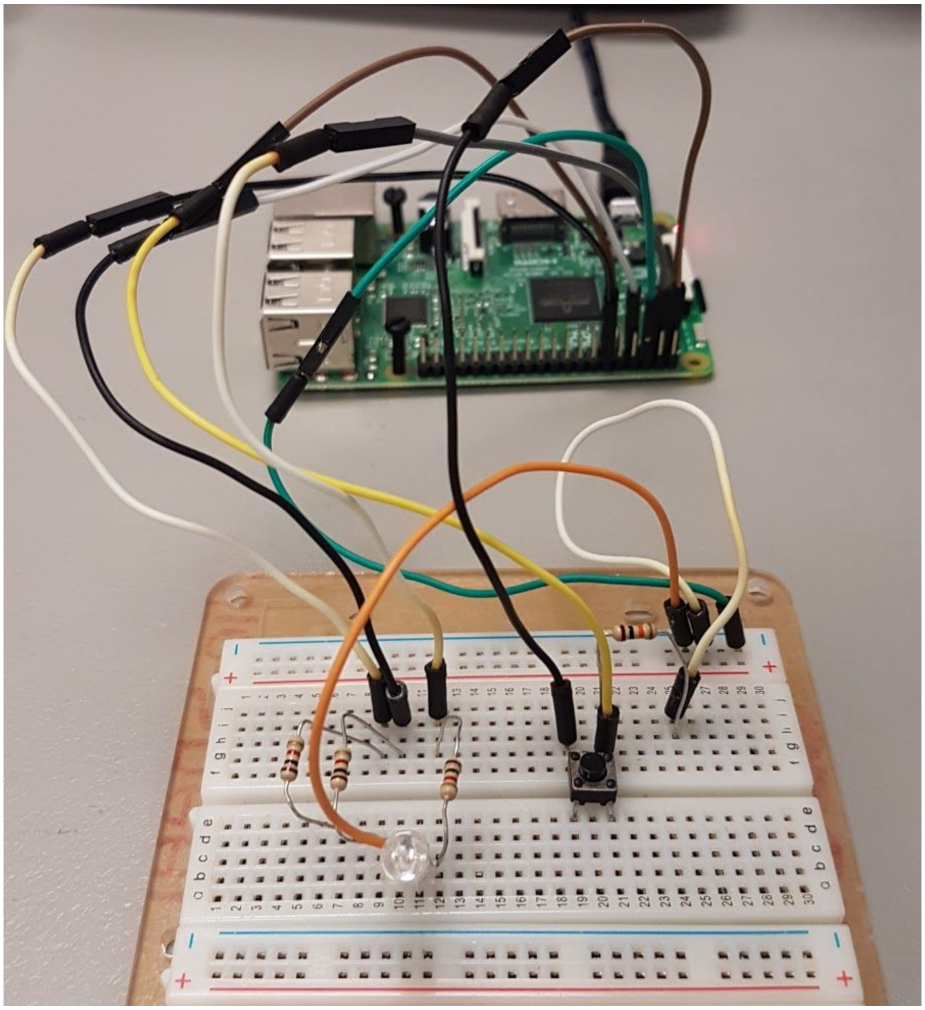
1) *Discuss what is the advantage/disadvantage using assembly over C/Python.*
One of the disadvantages of using assembly over a high-level language such as C or Python is how hard it is to maintain it, especially for large, complex programs. Being able to debug assembly is more difficult than C/Python considering everything is related to data registers and memory locations, with no abstract variables to use as reference. Another disadvantage of assembly is that it is dependent on the system architecture, which means that a functional assembly code with one computer could fail in another computer and need to be rewritten. Despite programs in C/Python being more flexible and perhaps easier to develop due to being able to maintain and debug them more easily, an advantage of assembly is that it often has better performance. As a result, assembly can at times run faster, use less memory, and as a result have a lower cost. The reason for this is that a compiler will have a difficult time providing an optimized code if the program in C/Python is really complex, using more instructions and steps to reach the final solution. Assembly, on the other hand, will be able to execute faster as it knows how to delegate the work directly.

2) *We have used a loop to keep checking on the signal from the input source(s). Is there an alternative way to read input from external button or switch than simply pulling the signal?*
The input from the button/switch can be read without pulling the signal by using interrupts. This would involve using particular functions that come with the RPi module such as GPIO.wait_for_edge or GPIO.add_event_detect and GPIO.add_event_callback. GPIO.wait_for_edge will simply wait until either a rising edge or falling edge is detected by the interrupt handler. From here, the code can resume with whatever the user wanted to happen when the button was pressed. GPIO.add_event_detect works similarly as the condition to wait for is passed through the function, and then the callback is set by using GPIO.add_event_callback. The advantage of using this method over continuously checking the pin's status in a while loop is that it doesn't demand as much processing power.

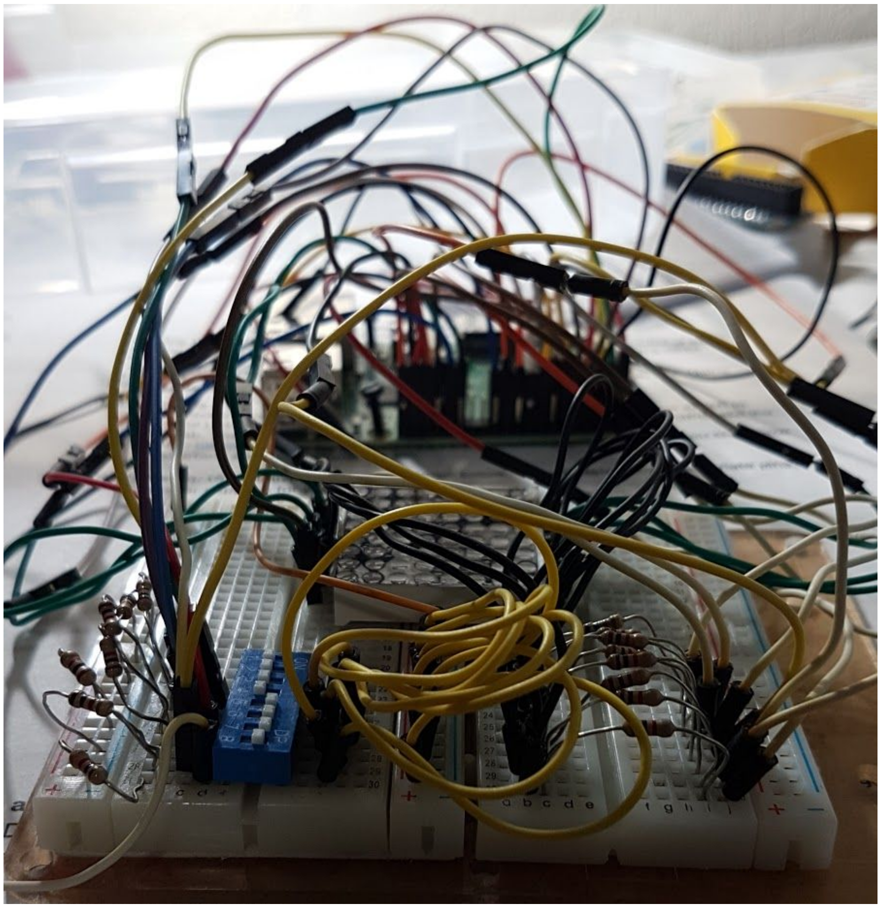
3) *If you replace the input switch by a temperature sensor, can you read temperature value from the sensor same way as you did from the switch? If yes, explain. If no, provide a solution to correctly read data from a temperature sensor TMP36.*
The temperature sensor TMP36 actually produces an analog signal that is directly proportional to temperature. However, the Raspberry Pi (RPi) is actually not capable of reading signals or data from an analog device. This is due to the lack of an analog to digital converter circuit. As a result, being able to read in the data from the temperature sensor will not be the same and will need additional hardware components in order to

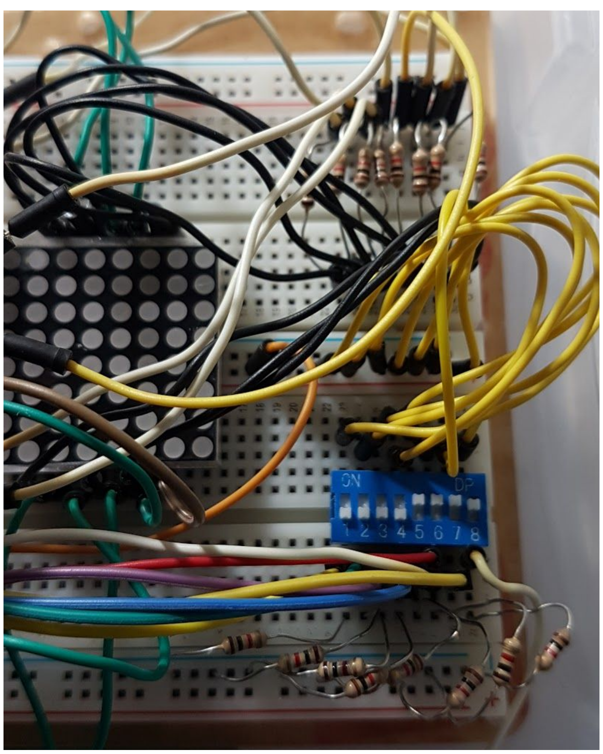
provide an analog device interface circuit for the RPi. There are multiple ways this can be accomplished, including the easiest way which is to wire an ADC chip to the RPi, such as MCP3008 chip. This will easily allow you to connect the chip to the GPIO pins in RPi and allow you to use various channels in order to read the analog inputs from the sensor. Another alternative to reading the analog input sensor is creating a charging-discharging circuit. This can be done by connecting a resistor and a capacitor in series and connecting those wires with power (3.3 V) and ground of the GPIO pins and coding the pins to go from LOW to HIGH in order to register a reading. This is basically a step response technique that responds to an electrical pulse that transitions from low to high within a specific time. As the cycle continues charging and discharging, the Python code can be used to provide the continuous value of the analog device's data being read by the RPi. As a result, we will be able to read the analog signal from the temperature sensor.

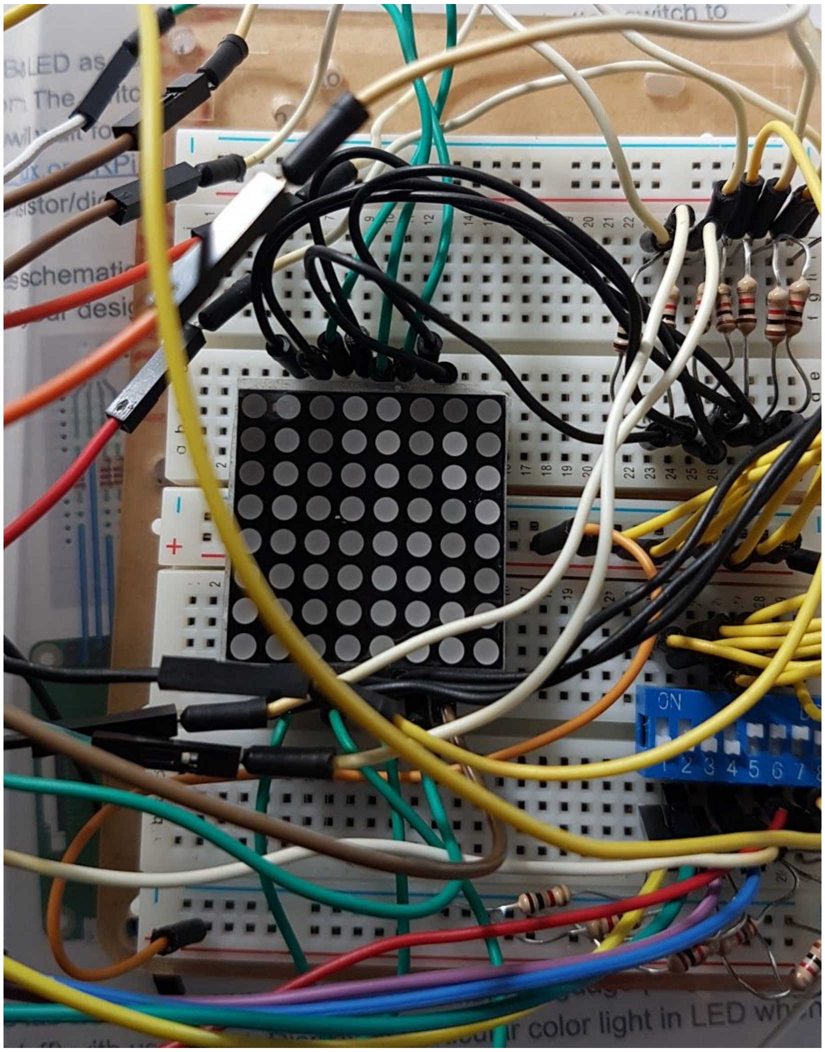
RGB LED Lab Activity 1: Working Circuit Design



LED Matrix and Dip Switch: Working Circuit Design







```
            .data
            .balign 4
red:        .int 3
green:      .int 7
blue:       .int 12
button:     .int 11
delay_time: .int 1000
error:      .asciz "Error in intitializing\n"
OUTPUT      = 1
INPUT       = 0

before_loop:.asciz "Before loop\n"
during_loop:.asciz "During Loop: %d\n"
r5_value:   .asciz "VALUE: %d\n"
result:     .word 0

            .text
            .global main
            .extern printf
            .extern wiringPiSetupPhys
            .extern pinMode
            .extern digitalWrite
            .extern digitalRead
            .extern delay

main:
    push    {r12, lr}
    bl      wiringPiSetupPhys
    mov     r1, #-1
    cmp     r0, r1
    bne     init
    ldr     r0, =error
    bl      printf
    b       exit


init:
    ldr     r0, =button
    ldr     r0, [r0]
    mov     r1, #INPUT
    bl      pinMode

    ldr     r4, =red
    ldr     r4, [r4]
    mov     r6, #1

@While loop
loop:
@digitalRead Setup for (pin)
    ldr     r0, =button
    ldr     r0, [r0]
    bl      digitalRead
    cmp     r0, r6
    bne     loop

@PinMode Setup for Light
lighton:
    mov     r0, r4
    mov     r1, #OUTPUT
    bl      pinMode
@digitalWrite Setup for (pin, 1)
    mov     r0, r4
    mov     r1, #1
    bl      digitalWrite

@delay(200)
```

```
    ldr     r0, =delay_time
    ldr     r0, [r0]
    bl      delay
@digitalWrite Setup for (pin, 0)
    mov     r0, r4
    mov     r1, #0
    bl      digitalWrite
colorLED:
    cmp     r4, #3
    beq     changeGreen
    cmp     r4, #7
    beq     changeBlue
    cmp     r4, #12
    beq     changeRed
changeGreen:
    ldr     r4, =green
    ldr     r4, [r4]
    b       loop
changeBlue:
    ldr     r4, =blue
    ldr     r4, [r4]
    b       loop
changeRed:
    ldr     r4, =red
    ldr     r4, [r4]
    b       loop
exit:
    pop {r12, pc}
```

```
# -*- coding: utf8 -*-
import RPi.GPIO as IO  #calling for header file which helps in using GPIO’s of PI
import time            #calling for time to provide delays in program
IO.setwarnings(False)  #do not show any warnings
x=1
y=1

#Initialize the pins for Matrix LEDs
IO.setmode (IO.BCM)  #programming the GPIO by BCM pin numbers. (like PIN29 as'GPIO5')
IO.setup(12,IO.OUT)  #initialize GPIO12 as an output.
IO.setup(22,IO.OUT)  #initialize GPIO22 as an output.
IO.setup(27,IO.OUT)
IO.setup(25,IO.OUT)
IO.setup(17,IO.OUT)
IO.setup(24,IO.OUT)
IO.setup(23,IO.OUT)
IO.setup(18,IO.OUT)
IO.setup(21,IO.OUT)
IO.setup(20,IO.OUT)
IO.setup(26,IO.OUT)
IO.setup(16,IO.OUT)
IO.setup(19,IO.OUT)
IO.setup(13,IO.OUT)
IO.setup(6,IO.OUT)
IO.setup(5,IO.OUT)

#Initialize the pins for DIP Switch
IO.setup(2,IO.IN)  #initialize GPIO2 (PIN 3) as an output.
IO.setup(3,IO.IN)  #initialize GPIO3 (PIN 5) as an output.
IO.setup(4,IO.IN)
IO.setup(14,IO.IN)
IO.setup(15,IO.IN)
IO.setup(10,IO.IN)
IO.setup(9,IO.IN)
IO.setup(11,IO.IN)

PORTVALUE = [128,64,32,16,8,4,2,1]
#value of pin in each port
A=[0,0b01111111,0b11111111,0b11001100,0b11001100,0b11001100,0b11111111,0b01111111]
B =[0,0b00111100,0b01111110,0b11011011,0b11011011,0b11011011,0b11111111,0b11111111]
C= [0,0b11000011,0b11000011,0b11000011,0b11000011,0b11100111,0b01111110,0b00111100]
D=[0,0b01111110,0b10111101,0b11000011,0b11000011,0b11000011,0b11111111,0b11111111]
E=[0,0b11011011,0b11011011,0b11011011,0b11011011,0b11011011,0b11111111,0b11111111]
F=[0,0b11011000,0b11011000,0b11011000,0b11011000,0b11011000,0b11111111,0b11111111]
G=[0b00011111,0b11011111,0b11011000,0b11011011,0b11011011,0b11011011,0b11111111,0b11111111]
H=[0,0b11111111,0b11111111,0b00011000,0b00011000,0b00011000,0b11111111,0b11111111]
I=[0b11000011,0b11000011,0b11000011,0b11111111,0b11111111,0b11000011,0b11000011,0b11000011]
J=[0b11000000,0b11000000,0b11000000,0b11111111,0b11111111,0b11000011,0b11001111,0b11001111]
K=[0,0b11000011,0b11100111,0b01111110,0b00111100,0b00011000,0b11111111,0b11111111]
L=[0b00000011,0b00000011,0b00000011,0b00000011,0b00000011,0b00000011,0b11111111,0b11111111]
M=[0b11111111,0b11111111,0b01100000,0b01110000,0b01110000,0b01100000,0b11111111,0b11111111]
N=[0b11111111,0b11111111,0b00011100,0b00111000,0b01110000,0b11100000,0b11111111,0b11111111]
O=[0b01111110,0b11111111,0b11000011,0b11000011,0b11000011,0b11000011,0b11111111,0b01111110]
P=[0,0b01110000,0b11111000,0b11001100,0b11001100,0b11001100,0b11111111,0b11111111]
Q=[0b01111110,0b11111111,0b11001111,0b11011111,0b11011011,0b11000011,0b11111111,0b01111110]
R=[0b01111001,0b11111011,0b11011111,0b11011110,0b11011100,0b11011000,0b11111111,0b11111111]
S=[0b11001110,0b11011111,0b11011011,0b11011011,0b11011011,0b11011011,0b11111011,0b0111001
```

```
1]
T=[0b11000000,0b11000000,0b11000000,0b11111111,0b11111111,0b11000000,0b11000000,0b11000000]
U=[0b11111110,0b11111111,0b00000011,0b00000011,0b00000011,0b00000011,0b11111111,0b11111110]
V=[0b11100000,0b11111100,0b00011110,0b00000011,0b00000011,0b00011110,0b11111100,0b11100000]
W=[0b11111110,0b11111111,0b00000011,0b11111111,0b11111111,0b00000011,0b11111111,0b11111110]
X=[0b01000010,0b11100111,0b01111110,0b00111100,0b00111100,0b01111110,0b11100111,0b01000010]
Y=[0b01000000,0b11100000,0b01110000,0b00111111,0b00111111,0b01110000,0b11100000,0b01000000]
Z=[0b11000011,0b11100011,0b11110011,0b11111011,0b11011111,0b11001111,0b11000111,0b11000011]

def PORT(pin):  #assigning GPIO state by taking 'pin' value
    if(pin&0x01 == 0x01):
        IO.output(21,0)   #if bit0 of 8bit 'pin' is true pull PIN21 low
    else:
        IO.output(21,1)   #if bit0 of 8bit 'pin' is false pull PIN21 high
    if(pin&0x02 == 0x02):
        IO.output(20,0)   #if bit1 of 8bit 'pin' is true pull PIN20 low
    else:
        IO.output(20,1)   #if bit1 of 8bit 'pin' is false pull PIN20 high
    if(pin&0x04 == 0x04):
        IO.output(26,0)   #if bit2 of 8bit 'pin' is true pull PIN26 low
    else:
        IO.output(26,1)   #if bit2 of 8bit 'pin' is false pull PIN26 high
    if(pin&0x08 == 0x08):
        IO.output(16,0)
    else:
        IO.output(16,1)
    if(pin&0x10 == 0x10):
        IO.output(19,0)
    else:
        IO.output(19,1)
    if(pin&0x20 == 0x20):
        IO.output(13,0)
    else:
        IO.output(13,1)
    if(pin&0x40 == 0x40):
        IO.output(6,0)
    else:
        IO.output(6,1)
    if(pin&0x80 == 0x80):
        IO.output(5,0)
    else:
        IO.output(5,1)


def PORTP(pinp):    #assigning GPIO logic for positive terminals by taking 'pinp' value
    if(pinp&0x01 == 0x01):
        IO.output(12,1)     #if bit0 of 8bit 'pinp' is true pull PIN12 high
    else:
        IO.output(12,0)     #if bit0 of 8bit 'pinp' is false pull PIN12 low
    if(pinp&0x02 == 0x02):
        IO.output(22,1)     #if bit1 of 8bit 'pinp' is true pull PIN22 high
    else:
        IO.output(22,0)     #if bit1 of 8bit 'pinp' is false pull PIN22 low
    if(pinp&0x04 == 0x04):
        IO.output(27,1)     #if bit2 of 8bit 'pinp' is true pull PIN27 high
    else:
        IO.output(27,0)     #if bit2 of 8bit 'pinp' is false pull PIN27 low
    if(pinp&0x08 == 0x08):
        IO.output(25,1)
```

```
    else:
        IO.output(25,0)
    if(pinp&0x10 == 0x10):
        IO.output(17,1)
    else:
        IO.output(17,0)
    if(pinp&0x20 == 0x20):
        IO.output(24,1)
    else:
        IO.output(24,0)
    if(pinp&0x40 == 0x40):
        IO.output(23,1)
    else:
        IO.output(23,0)
    if(pinp&0x80 == 0x80):
        IO.output(18,1) #if bit7 of 8bit 'pinp' is true pull PIN18 high
    else:
        IO.output(18,0) #if bit7 of 8bit 'pinp' is false pull PIN18 low


while True:
    full_bits = ""
    dip_1 = IO.input(2)
    dip_2 = IO.input(3)
    dip_3 = IO.input(4)
    dip_4 = IO.input(14)
    dip_5 = IO.input(15)
    dip_6 = IO.input(10)
    dip_7 = IO.input(9)
    dip_8 = IO.input(11)
    full_bits4 = full_bits + str(dip_1) + str(dip_2) + str(dip_3) + str(dip_4)
    full_bits8 = full_bits + str(dip_5) + str(dip_6) + str(dip_7) + str(dip_8)
    print full_bits
    hex_value4 = hex(int(full_bits4,2))
    hex_value8 = hex(int(full_bits4,2))
    print hex_value4
    print hex_value8
    time.sleep(1)

    for y in range (100):   #execute loop 100 times
        for x in range (8): #execute the loop 8 times incrementing x value from zero to
        seven
            pin  = PORTVALUE[x]  #assigning value to 'pin' for each digit
            PORT(pin);  #mapping appropriate GPIO
            pinp= hex_value4[x]  #assigning character of the first 4 bits
            PORTP(pinp); #turning the GPIO to show character of the first 4 bits
            pinp= hex_value8[x] #assigning character of the last 4 bits
            PORTP(pinp); #turning the GPIO to show character of the last 4 bits
            time.sleep(0.0005) #wait for 0.5msec

    pinp= 0
    PORTP(pinp);
    time.sleep(1)
```