Gevorg Markarov, Armen Jabamiko, Sarah Primavera

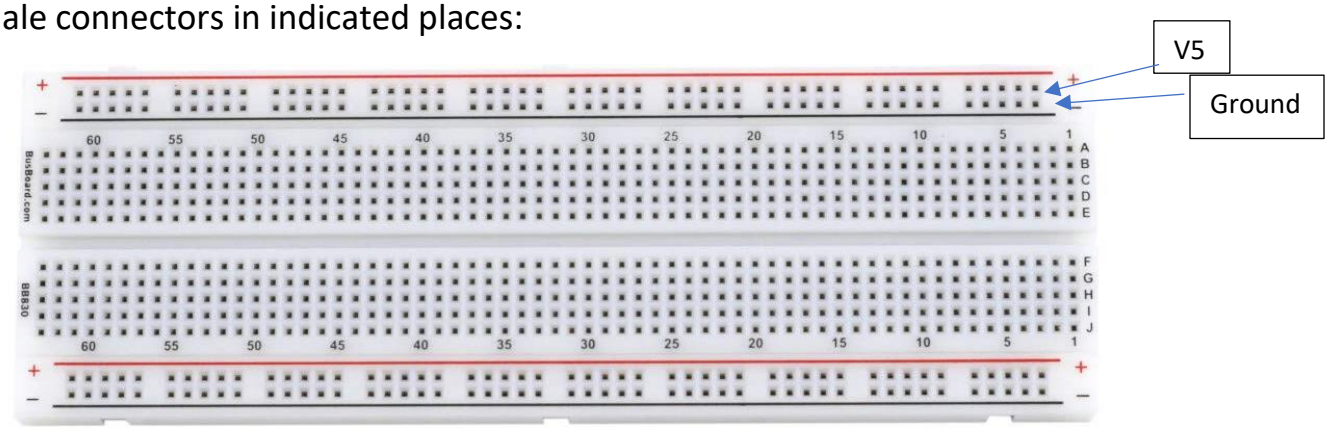
Unix

Section 2

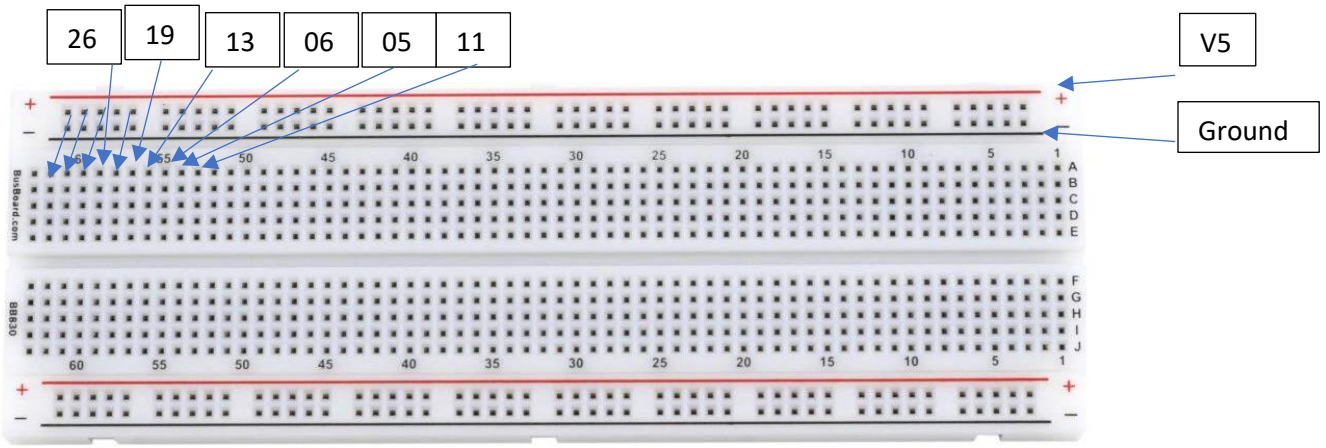
Red Light Green Light

The purpose of the project is to create a mechanism using Rosebery Pi to recreate the game “Red Light Green Light” specifically, the one from the show called “Squid Games”

At first, we will need to connect V5 and Ground pins to the breadboard by using female connectors in indicated places:



After we connected all the following ports to the LCD display: (All the numbers represent the corresponding one for a GPIO



And on the bottom, just skipping the first we should place the LCD display so it connects to all the connector

After, keeping everything on the breadboard we need to connect the Ground and GPIO4 to the green light and the same way, another Ground and GPIO27 to the red light the ground should be connected to the short connector of the light and the GPIO to the long one.

For the servo motor the ground should be connected to the negative connector of the servo motor, the V5 to the positive one and the GPIO (In our case 18th) should go to the pulse. If we look to the colors, the ground is connected to the black connector, V5 to the red one and GPIO to the yellow one.

In the end, for the code, we need to open the Ros pi connected to a monitor and that has a keyboard access (preferably a mouse access as well) and open the python/Thonny and copy past the following code there

```
from gpiozero import LED
from gpiozero import AngularServo
from time import sleep
import time
import random
import RPi.GPIO as GPIO

green = LED(4)
red = LED(27)

def turn180():
    servo = AngularServo(18, min_pulse_width=0.0005, max_pulse_width=0.0025)
    servo.angle = 90
    servo.angle = 90
    sleep(1.5)
    sleep(random.randint(2,4))
    red.off()
    green.on()
    servo.angle = -90
    servo.angle = -90
    sleep(1.5)
    servo.close()
LCD_RS = 26
LCD_E  = 19
LCD_D4 = 13
LCD_D5 = 6
LCD_D6 = 5
LCD_D7 = 11
```

```
LED_ON = 15

# Define some device constants
LCD_WIDTH = 16    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line

# Timing constants
E_PULSE = 0.00005
E_DELAY = 0.00005

def main():
  # Main program block

  # Initialise display
  lcd_init()

  # Toggle backlight on-off-on
  GPIO.output(LED_ON, True)

  # Send some centred test
  lcd_byte(LCD_LINE_1, LCD_CMD)
  gevorgString = "10"
  gevorg = 10
  green.off()
  red.on()
  while (gevorg >= 0):
      lcd_byte(LCD_LINE_1, LCD_CMD)
      gevorgString = str(gevorg)
      gevorg -= 1
      lcd_string("Steps left: "+gevorgString,2)
      value = random.randint(2,4)
      sleep(value)
      green.off()
      red.on()
      turn180()

  green.off()
  red.off()
  lcd_byte(LCD_LINE_2, LCD_CMD)
```

```
  lcd_string("GAME OVER",2)
  # Turn off backlight
  GPIO.output(LED_ON, False)

def lcd_init():
  GPIO.setmode(GPIO.BCM)       # Use BCM GPIO numbers
  GPIO.setup(LCD_E, GPIO.OUT)  # E
  GPIO.setup(LCD_RS, GPIO.OUT) # RS
  GPIO.setup(LCD_D4, GPIO.OUT) # DB4
  GPIO.setup(LCD_D5, GPIO.OUT) # DB5
  GPIO.setup(LCD_D6, GPIO.OUT) # DB6
  GPIO.setup(LCD_D7, GPIO.OUT) # DB7
  GPIO.setup(LED_ON, GPIO.OUT) # Backlight enable
  # Initialise display
  lcd_byte(0x33,LCD_CMD)
  lcd_byte(0x32,LCD_CMD)
  lcd_byte(0x28,LCD_CMD)
  lcd_byte(0x0C,LCD_CMD)
  lcd_byte(0x06,LCD_CMD)
  lcd_byte(0x01,LCD_CMD)

def lcd_string(message,style):
  # Send string to display
  # style=1 Left justified
  # style=2 Centred
  # style=3 Right justified

  if style==1:
    message = message.ljust(LCD_WIDTH," ")
  elif style==2:
    message = message.center(LCD_WIDTH," ")
  elif style==3:
    message = message.rjust(LCD_WIDTH," ")

  for i in range(LCD_WIDTH):
    lcd_byte(ord(message[i]),LCD_CHR)

def lcd_byte(bits, mode):
  # Send byte to data pins
  # bits = data
  # mode = True  for character
  #        False for command
```

```
        GPIO.output(LCD_RS, mode) # RS

        # High bits
        GPIO.output(LCD_D4, False)
        GPIO.output(LCD_D5, False)
        GPIO.output(LCD_D6, False)
        GPIO.output(LCD_D7, False)
        if bits&0x10==0x10:
          GPIO.output(LCD_D4, True)
        if bits&0x20==0x20:
          GPIO.output(LCD_D5, True)
        if bits&0x40==0x40:
          GPIO.output(LCD_D6, True)
        if bits&0x80==0x80:
          GPIO.output(LCD_D7, True)

        # Toggle 'Enable' pin
        time.sleep(E_DELAY)
        GPIO.output(LCD_E, True)
        time.sleep(E_PULSE)
        GPIO.output(LCD_E, False)
        time.sleep(E_DELAY)

        # Low bits
        GPIO.output(LCD_D4, False)
        GPIO.output(LCD_D5, False)
        GPIO.output(LCD_D6, False)
        GPIO.output(LCD_D7, False)
        if bits&0x01==0x01:
          GPIO.output(LCD_D4, True)
        if bits&0x02==0x02:
          GPIO.output(LCD_D5, True)
        if bits&0x04==0x04:
          GPIO.output(LCD_D6, True)
        if bits&0x08==0x08:
          GPIO.output(LCD_D7, True)

        # Toggle 'Enable' pin
        time.sleep(E_DELAY)
        GPIO.output(LCD_E, True)
        time.sleep(E_PULSE)
        GPIO.output(LCD_E, False)
        time.sleep(E_DELAY)
```

```
if __name__ == '__main__':
  main()
```

For the end, we need to make it work in a way so the code work whenever we boot the Raspberry Pi

Firstly, you will need to find the location of the python file.

After you will need to write “sundo crontab -e” in your terminal

It will open a file with text. There, in the bottom you will need to write

“@reboot python3” and the file’s path the following way “@reboot python3 /home/pi/folder/filename.py”

After save the file. Write in the terminal “sudo raspi-config”

Select the “Boot Options”, after select “Console Autologin Text console, automatically logged in as ‘pi’ user” and then try to reboot so you see the result.