Bobby Vielma CSCE 462

Alexia Perez 1/29/2021

Lab 1 Report

1. RPi CPU speed is 1200 when looking for the arm frequency. When looking at the CPU frequency we got 600,000. You can overclock the RPi to get higher Hertz, but there can be potential consequences such as the RPi getting too hot, the program not running, the program slowing down, and Raspbian refusing to start. Any of these signs means you need to slow down the speeds.
2. From what I’ve gathered, whenever a pin is set to logic low it will fall between 0v and 0.8v. This is for the BCM2835
3. When a pin is set to logic high it will fall between 1.3v and 3.3v. Again, this is for BCM2835
   1. Technically you can use the Raspberry Pi 3 as a PC. It does have web capabilities, HDMI ports, USB ports, audio jacks, and other accessories. It may run slower in some respects, especially when it comes to the regular apps many people use (Gmail, Google Drive, etc.). Not everything may be smooth at first so some tweaks or workarounds may be necessary. The memory storage is also smaller
   2. The main application of the RPi is to teach people how their computers really work behind the scenes. Further applications are in things like tablet computers, home automation, and internet radio. Because of its small size the RPi can be used in many applications doesn’t quite suite a traditional pc
   3. Polling is when the program is kept running (i.e. while-loop) and the OS is waiting on some sort of input from the user. Essentially, it’s the OS asking the program if it is ready continuously. This burns CPU power even when the program isn’t doing anything. An interrupt is when the program interrupts the OS and grabs its attention. Instead of having the OS continuously ask the program if it’s ready, it can use those resources towards something else and when the program notifies that it is ready only then does the OS spend its resources.
   4. Polling is good whenever there is a short amount of time that the OS must continuously ask if the program is ready. If the button was pressed every 5 seconds, then polling may be a good idea as there isn’t a long time between button presses. Interrupts are good if there is a chance at long periods of times before the signal (button press in our case) is received. Someone could leave the program running for 20 minutes before deciding to press the button. In that case, interrupts are preferable.
4. OS
   1. Hardware interrupts are generated from an external tool or device. They can be maskable or non-maskable, are asynchronous, and have a lower priority than software interrupts. They can be generated by an outside device requesting to start an I/O process (e.g. mouse clicks and keyboard presses).
   2. Software interrupts are generated by an internal system of the computer. They can be normal interrupts or exceptions, are synchronous, highest priority of all interrupts, and increments the program counter. Examples are system calls and during error and exception handling.

**Polling Code**

**from** gpiozero **import** LED, Button  
**from** time **import** sleep  
**import** time  
**import** threading  
**import** RPi.GPIO **as** GPIO  
**import** sys  
  
GPIO.setmode(GPIO.BCM)  
  
*# Global Variables*elapsed = sys.maxint  
startTime = 0  
  
*# Traffic Light 2*red2 = LED(4)  
green2 = LED(5)  
blue2 = LED(6)  
  
*#Traffic Light 1*red = LED(20)  
green = LED(13)  
blue = LED(19)  
  
*# Seven Segment Display*pins = [18, 23, 26, 27, 22, 12, 25]  
  
GPIO.setup(18, GPIO.OUT) *# Intialize GPIO Pins as output*GPIO.setup(23, GPIO.OUT)  
GPIO.setup(26, GPIO.OUT)  
GPIO.setup(27, GPIO.OUT)  
GPIO.setup(22, GPIO.OUT)  
GPIO.setup(12, GPIO.OUT)  
GPIO.setup(25, GPIO.OUT)  
GPIO.setup(21, GPIO.OUT) *# dot*arrSequence = [[1,1,1,1,1,1,0], *# 0* [0,1,1,0,0,0,0], *# 1* [1,1,0,1,1,0,1], *# 2* [1,1,1,1,0,0,1], *# 3* [0,1,1,0,0,1,1], *# 4* [1,0,1,1,0,1,1], *# 5* [1,0,1,1,1,1,1], *# 6* [1,1,1,0,0,0,0], *# 7* [1,1,1,1,1,1,1], *# 8* [1,1,1,0,0,1,1] *# 9* ]  
  
*# Button*button = Button(17)  
   
**def** blink():  
 **while** countdown > 0:  
 blue.on()  
 sleep(0.3)  
 blue.off()  
 sleep(0.3)  
  
**try**:  
 **while** True:  
 green2.on() *# Traffic Light 2 starts green* **if** elapsed < 20: *# Make sure to wait 20 seconds before button can be pressed* elapsed = time.time() - startTime  
 **continue  
   
 if** button.is\_pressed **and** elapsed >= 20:  
 startTime = time.time() *# Get starting time of pressing button  
   
 # green2.off()* sleep(1)  
 **for** i **in** range(3): *# blink blue three times* blue2.on()  
 sleep(1)  
 blue2.off()  
 sleep(0.5)  
   
 red2.on() *# Turn red on and wait for countdown to reach zero* green.on() *# Traffic Light 1 turns green* countdown = 9  
 **while** countdown >= 0:  
 **for** i **in** range(7): *# get number sequence* GPIO.output(pins[i], arrSequence[countdown][i]) *# activate pin[i] on high or low (0 - 1) depending on arrSequence* sleep(1)  
   
 **if** countdown == 5: *# If countdown is less than 5* green.off()  
 *#if countdown == 4:* t1 = threading.Thread(target=blink)  
 t1.start()  
   
 countdown -= 1 *# Decrement the countdown shown* red2.off()  
 red.on() *# After timer Traffic Light 1 becomes red and* green2.on() *# Traffic Light 2 becomes green* sleep(1)  
 red.off()  
 green2.off()  
   
 elapsed = time.time() - startTime  
   
   
**except** KeyboardInterrupt:  
 GPIO.cleanup() *# clean up GPIO on Ctrl+C exit*GPIO.cleanup() *# clean up at end of program*

**Interrupt Code**

**from** gpiozero **import** LED, Button  
**from** time **import** sleep  
**import** time  
**import** threading  
**import** RPi.GPIO **as** GPIO  
**from** signal **import** pause  
**import** sys  
  
  
GPIO.setmode(GPIO.BCM)  
  
*# Traffic Light 2 using GPIOzero library*red2 = LED(4)  
green2 = LED(5)  
blue2 = LED(6)  
  
*#Traffic Light 1 using GPIOzero library*red = LED(20)  
green = LED(13)  
blue = LED(19)  
  
*# Seven Segment Display*pins = [18, 23, 26, 27, 22, 12, 25]  
  
GPIO.setup(18, GPIO.OUT) *# Intialize GPIO Pins as output*GPIO.setup(23, GPIO.OUT)  
GPIO.setup(26, GPIO.OUT)  
GPIO.setup(27, GPIO.OUT)  
GPIO.setup(22, GPIO.OUT)  
GPIO.setup(12, GPIO.OUT)  
GPIO.setup(25, GPIO.OUT)  
GPIO.setup(21, GPIO.OUT) *# dot*arrSequence = [[1,1,1,1,1,1,0], *# 0* [0,1,1,0,0,0,0], *# 1* [1,1,0,1,1,0,1], *# 2* [1,1,1,1,0,0,1], *# 3* [0,1,1,0,0,1,1], *# 4* [1,0,1,1,0,1,1], *# 5* [1,0,1,1,1,1,1], *# 6* [1,1,1,0,0,0,0], *# 7* [1,1,1,1,1,1,1], *# 8* [1,1,1,0,0,1,1] *# 9* ]  
  
*# Button  
# GPIOzero library*button = Button(17)   
  
*# RPi.GPIO way of doing it. GPIO 17 set as input. Pulled up to stop false signals  
# GPIO.setup(17, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)  
  
# Makes a light blink blue. Note: This functionality can be reached with the blink() function  
# from the GPIOzero library***def** blink(countdown):  
 **while** countdown > 0:  
 blue.on()  
 sleep(0.3)  
 blue.off()  
 sleep(0.3)  
 countdown -=1  
*# END OF BLINK  
  
# Function for when button is pressed***def** pressed():  
 **try**:  
 startTime = time.time()  
   
 *# GPIO.wait\_for\_edge(17, GPIO.FALLING) # RPi.GPIO way of doing it* green2.off()  
 sleep(1)  
   
 **for** i **in** range(3): *# blink blue three times* blue2.on()  
 sleep(1)  
 blue2.off()  
 sleep(0.5)  
   
 red2.on() *# Turn red on and wait for countdown to reach zero* green.on() *# Traffic Light 1 turns green* countdown = 9  
 **while** countdown >= 0:  
 **for** i **in** range(7): *# get number sequence* GPIO.output(pins[i], arrSequence[countdown][i]) *# activate pin[i] on high or low (0 - 1) depending on arrSequence* sleep(0.7)  
   
 **if** countdown == 5: *# If countdown is less than 5* green.off()  
 t1 = threading.Thread(target=blink, args=(countdown,)) *# Create new thread to blink first light blue* t1.start()  
   
 countdown -= 1 *# Decrement the countdown shown* red2.off()  
 red.on() *# After timer Traffic Light 1 becomes red and* green2.on() *# Traffic Light 2 becomes green* sleep(1)  
 red.off()  
 green2.off()  
   
 **while** time.time() - startTime < 19.5:  
 **continue** green2.on()   
   
 **except** KeyboardInterrupt:  
 GPIO.cleanup() *# clean up GPIO on Ctrl+C exit  
 # GPIO.cleanup()  
# END OF PRESSED*green2.on() *# start program with Traffic Light 2 green*button.when\_pressed = pressed *# Run pressed() when the button is pressed  
  
   
# pause() # pauses the script so it can wait for the button press***while** True:  
 time.sleep(1)