Byeongchan Gwak, Student ID : 501026

Washington University in St.Louis

McKelvey School of Engineering

Fall Semester 2021

CSE467M: Embedded Computing Systems

Homework #5

Chapter 6 problems:

1) **Q6-6 (15 points)**

CPU Utilization = (CPU time for useful work) / (Total available CPU time)

a. P1: period = 1s = 1000ms exec_time = 10ms, P2: period = 100ms exec_time = 10ms
   A. 10ms/1000ms + 10ms/100ms= 0.11
b. P1: period = 100ms exec_time = 25ms, P2: period = 80ms, exec_time = 15ms, P3: period = 40ms exec_time = 5ms
   A. 25ms/100ms + 15ms/80ms + 5ms/40ms = 0.25 + 0.1875 + 0.125 = 0.5625
c. P1: period = 10ms exec_time = 1ms, P2: period = 1ms exec_time = 0.2ms, P3: period = 0.2ms exec_time = 0.05ms
   A. 1ms/10ms + 0.2ms/1ms + 0.05ms/0.2ms = 0.1 + 0.2 + 0.25 = 0.55

2) **Q6-9 (10 points)**

Any process that can be executed is in the ready state. Process may not be ready to run waiting for something to happen first (e.g.g. a process is waiting for from another I/O device or another process)

3) **Q6-12 (15 points)**

a. The least common multiple of the period (LCM) is 10.

Because P1 = 2, P2 = 5, P3 = 10

b. The least common multiple of the period (LCM) is 20.

Because P1 = 2, P2 = 4, P3 = 5, P4 = 10

c. The least common multiple of the period (LCM) is 60.

Because P1 = 3, P2 = 4, P3 = 5, P4 = 6, P5 = 10

Byeongchan Gwak, Student ID : 501026

4) Q6-13 (15 points)

First, we need to calculate the current system's availability.

Process	Execution time	Deadline	Executed count (LCM=200)	Units of CPU time
P1	4	200	1	4*1 = 4
P2	1	10	20	1*20 = 20
P3	2	40	5	2*5 = 10
P4	6	50	4	6*4 = 24
			Total units of CPU time	4 + 20 + 10 + 24 = 58 < 200

The total unis of CPU time shows that it doesn't exceed available CPU capacity.

Let's add another instance of P1 to the system and re-calculate.

Process	Execution time	Deadline	Executed count (LCM=200)	Units of CPU time
P1 * 2	4	200	1	4*1 *2 = 8
P2	1	10	20	1*20 = 20
P3	2	40	5	2*5 = 10
P4	6	50	4	6*4 = 24
			Total units of CPU time	4 + 20 + 10 + 24 = 62 < 200

The total unis of CPU time still shows that it doesn't exceed available CPU capacity.

As a result, we can add another instance of P1 to the system.

5) Q6-14 (15 points)

Process	Execution time	Deadline	Executed count (LCM=100)	Units of CPU time
P1	1	10	10	1*10 = 10
P2	3	25	4	3*4 = 12
P3	X	50	2	X*2 = 2X
P4	10	100	1	10*1 = 10
			Total units of CPU time	10 + 12 + 2X + 10 <= 100 X <= 34

As a result, x can be a maximum of 34.

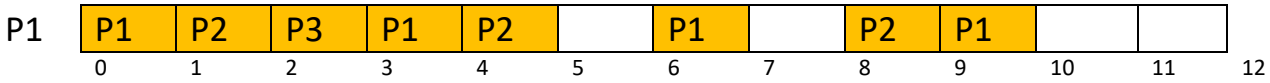
Byeongchan Gwak, Student ID : 501026

6) Q6-17 (30 points)

Process	Execution time	Deadline
P1	1	3
P2	1	4
P3	1	12

a. Schedule the processes using an RMS policy.

LCM = 12



b. Schedule the processes using an EDF policy.

Time	Running process	Deadlines
0	P1	
1	P2	
2	P3	P1
3	P1	P2
4	P2	
5		P1
6	P1	
7		P2
8	P2	P1
9	P1	
10		
11		P2, P3

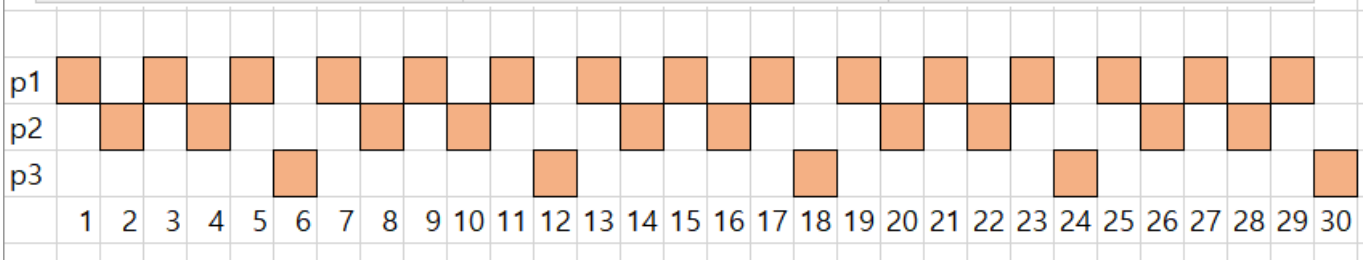
Byeongchan Gwak, Student ID : 501026

7) Q6-19 (30 points)

a. Schedule the processes using an RMS policy.

LCM = 30

Process	Execution time	Deadline
P1	1	2
P2	1	3
P3	2	10



b. Schedule the processes using an EDF policy.

Time	Running process	Deadlines
0	P1	
1	P2	P1
2	P1	P2
3	P2	P1
4	P1	
5	P3	P1, P2
6	P3	
7	P1	P1
8	P1	P2
9	P2	P1, P3
10	P1	
11	P2	P1, P2
12	P1	
13	P2	P1
14	P1	P2
15	P2	P1
16	P1	
17	P3	P1, P2
18	P3	
19	P1	P1, P3
20	P1	P2

Byeongchan Gwak, Student ID : 501026

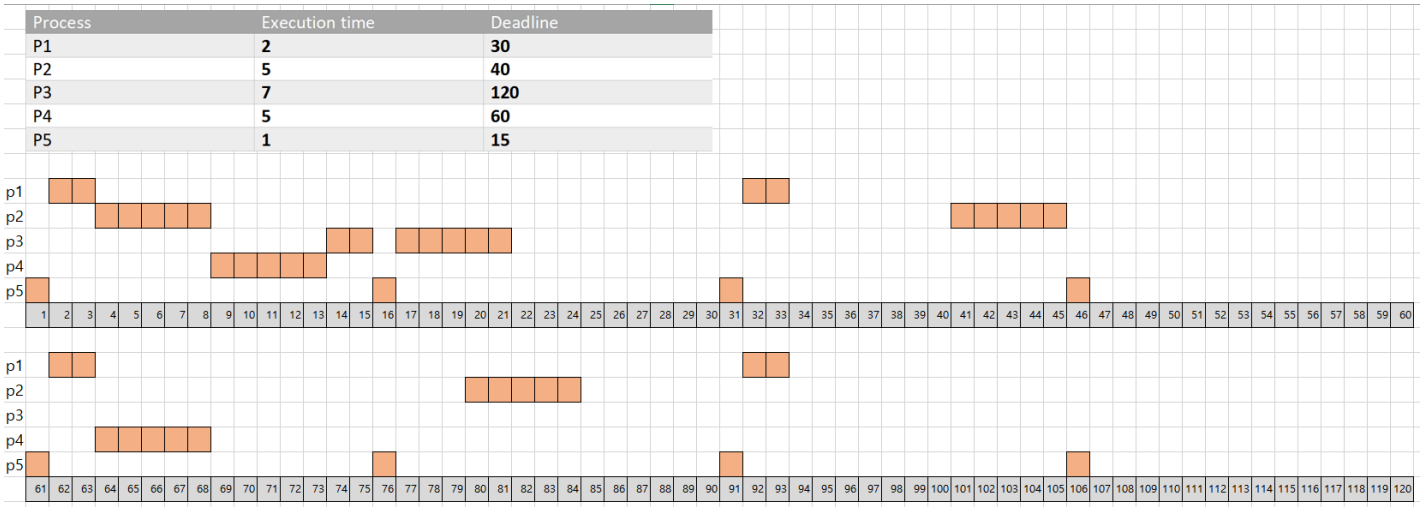
21	P2	P1
22	P1	
23	P2	P1, P2
24	P1	
25	P3	P1
27	P3	P2
28	P1	P1
29	P1	

Byeongchan Gwak, Student ID : 501026

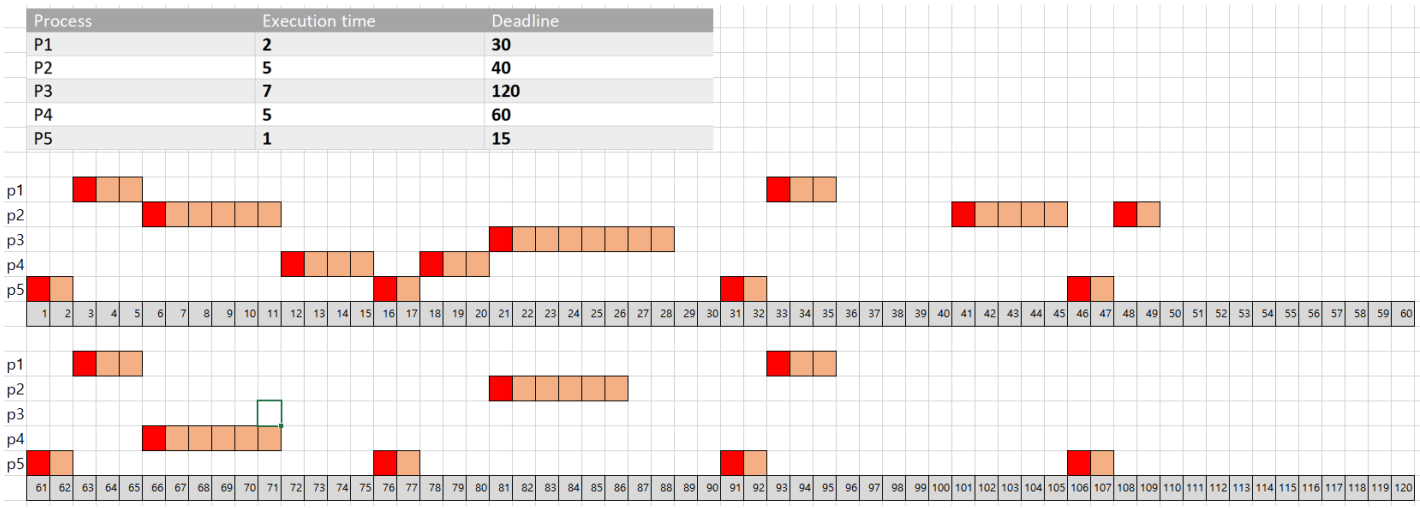
8) Q6-21 (30 points)

Process	Execution time	Deadline
P1	**2**	**30**
P2	**5**	**40**
P3	**7**	**120**
P4	**5**	**60**
P5	**1**	**15**

a. using standard RMS



b. adding one unit of overhead for each context switch



Byeongchan Gwak, Student ID : 501026

9) **Q6-24 (20 points)**

The RTOS is going to use 'Priority inheritance' to increase P2's priority to P1's level so as to P2 can finish the critical section. After finish the critical section and then reduce P2's priority back to its original level and then P1 can start again.

10) **Q6-25 (10 points)**

By dividing interrupt handling into two different pieces of code can reduce hardware interrupt effects.

- Interrupt service handler (ISH) – simple piece of code that performs the minimal operations required to respond to the device.
- Interrupt service routine (ISR) – performs the rest of the interrupt routine. ISR runs as a thread which allows the RTOS to ensure that all the tasks in the system receive their required resources

11) **Q6-26 (10 points)**

Dual-kernel approach where there are standard kernel for non-real time processes and specialized kernel (cokernel) for real-time processes (e.g. interrupts)

12) Raspberry Pi Circuit Card Assembly (CCA) Laboratory #3– Setting up serial communication between Raspberry Pi and Personal Computer (800 points)

12.1)

Use GPIO serial port on Raspberry Pi 3 Model B+ to make serial communication between Raspberry Pi and Personal Computer (PC).

You should be able to read and write messages on both PC and Raspberry Pi Card.

This is a very useful and practical exercise that you may use a lot when testing embedded systems by connecting them using common serial communication link to the PC. In this way you can monitor and test the embedded system.

For this lab exercise you will need to obtain USB-serial cable that you can get from here:

https://www.amazon.com/gp/product/B06ZYPLFNB/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&tag=scribles-20&camp=1789&creative=9325&linkCode=as2&creativeASIN=B06ZYPLFNB&linkId=4b3ec322c8753b76594c0f558fc119f7

You can follow this example exercise to make it work on your setup:

https://scribles.net/setting-up-serial-communication-between-raspberry-pi-and-pc/

Use this exact setup to make serial connection between the **Raspberry Pi and the PC.**

Make sure that RX connects to TX on one side to other.

12.2)

Use Python as shown below, to write continuously message "CSE467 Embedded Computing Systems" to the PC.

https://pimylifeup.com/raspberry-pi-serial/

Also use C as shown below, to write continuously message "CSE467 Embedded Computing Systems" to the PC.

https://www.electronicwings.com/raspberry-pi/raspberry-pi-uart-communication-using-python-and-c

12.3)

Write a report where you will show all your hardware and software setups for the part 12.1. Also submit your code for the part 12.2 and explain how it works. Provide a few photos of the PC display that shows the massage "CSE467 Embedded Computing Systems" and also photos of that show Raspberry Pi and PC displays from part 12.1.

12.4)

Submit a short video where you show that you can read and write messages on both Raspberry Pi Card and PC from part 12.1 and that you can do the part 12.2 as well by showing PC displaying the message "CSE467 Embedded Computing Systems".

- Please see the next page.

Byeongchan Gwak, Student ID : 501026

- Connection between the Pi and the Laptop for 12.1 & 12.2

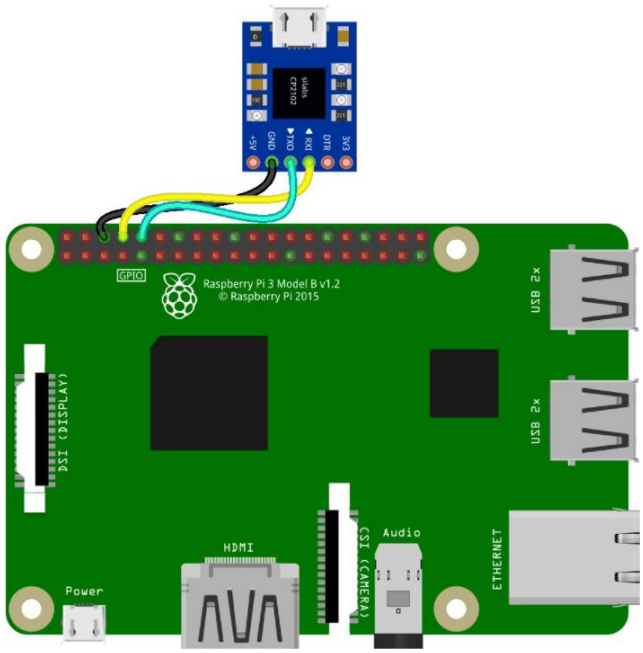
I made a huge mistake connecting UART pin on Pi, I connected USB-Serial cable Tx with GPIO's Tx and Rx with GPIO's Rx. I had to make a crossroads connection to send and receive data. I did it again and they can send and receive the data finally.

USB-Serial Cable GND(Pin1) with PI' GPIO 6

USB-Serial Cable TXD(Pin4) with PI' GPIO 15(UART RX)

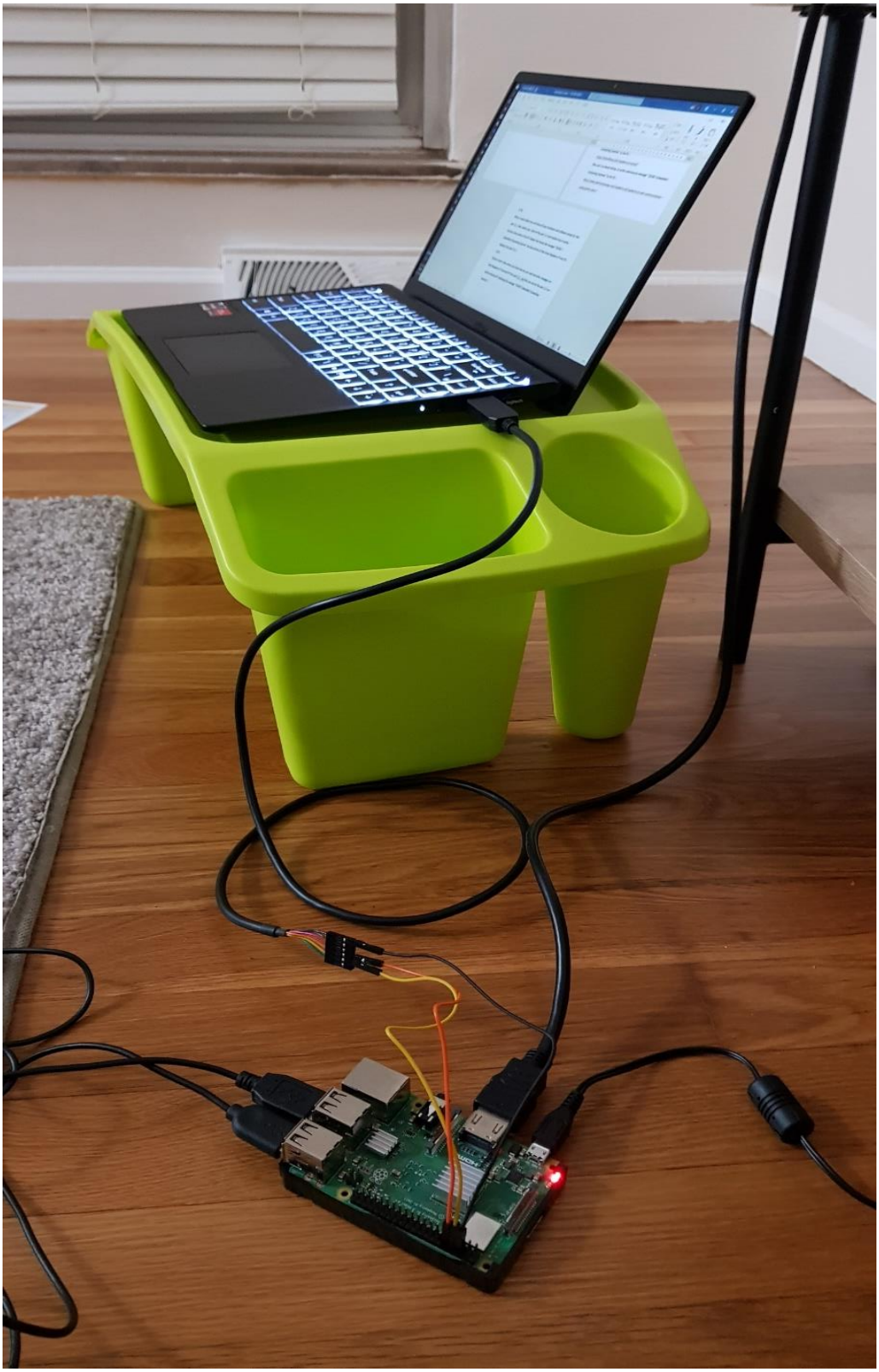
USB-Serial Cable RXD(Pin5) with PI' GPIO 14(UART TX)

Circuit



Source: https://electropeak.com/learn/raspberry-pi-serial-communication-uart-w-arduino-pc/

Byeongchan Gwak, Student ID : 501026



Byeongchan Gwak, Student ID : 501026

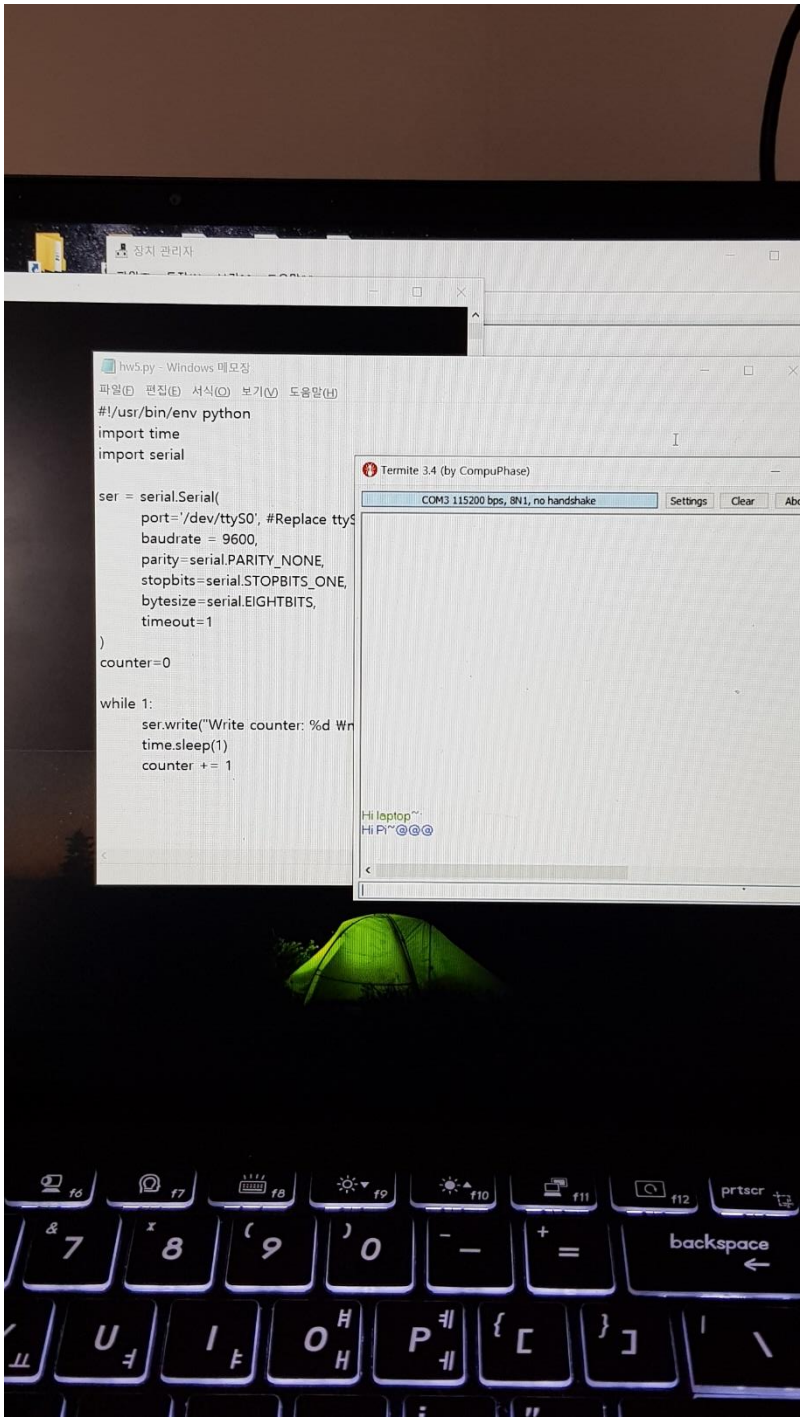
- **Source code**

```
# Byeongchan Gwak, Student ID : 501026
# This serial library contains about serial connections.
# It allows reading and writing through the serial ports.
import serial
import time

# Making a serial connection to write out through serial connection.
ser = serial.Serial(
        port='/dev/ttyS0',              # Define the serial port to read and write
        baudrate = 115200,              # I made a baudrate setting to 115200
        parity=serial.PARITY_NONE,      # For parity checking, but I set it 'No'
        stopbits=serial.STOPBITS_ONE,   # Indicates the end of the data transmission
        bytesize=serial.EIGHTBITS,      # The number of data bits
        timeout=1                       # The amount of time for before timing out
)

cmd='CSE467 Embedded Computing Systems. bcgwak\n'

while 1:
        # Write the cmd string value trough serial
        ser.write(cmd.encode())
        time.sleep(1)
```

Byeongchan Gwak, Student ID : 501026

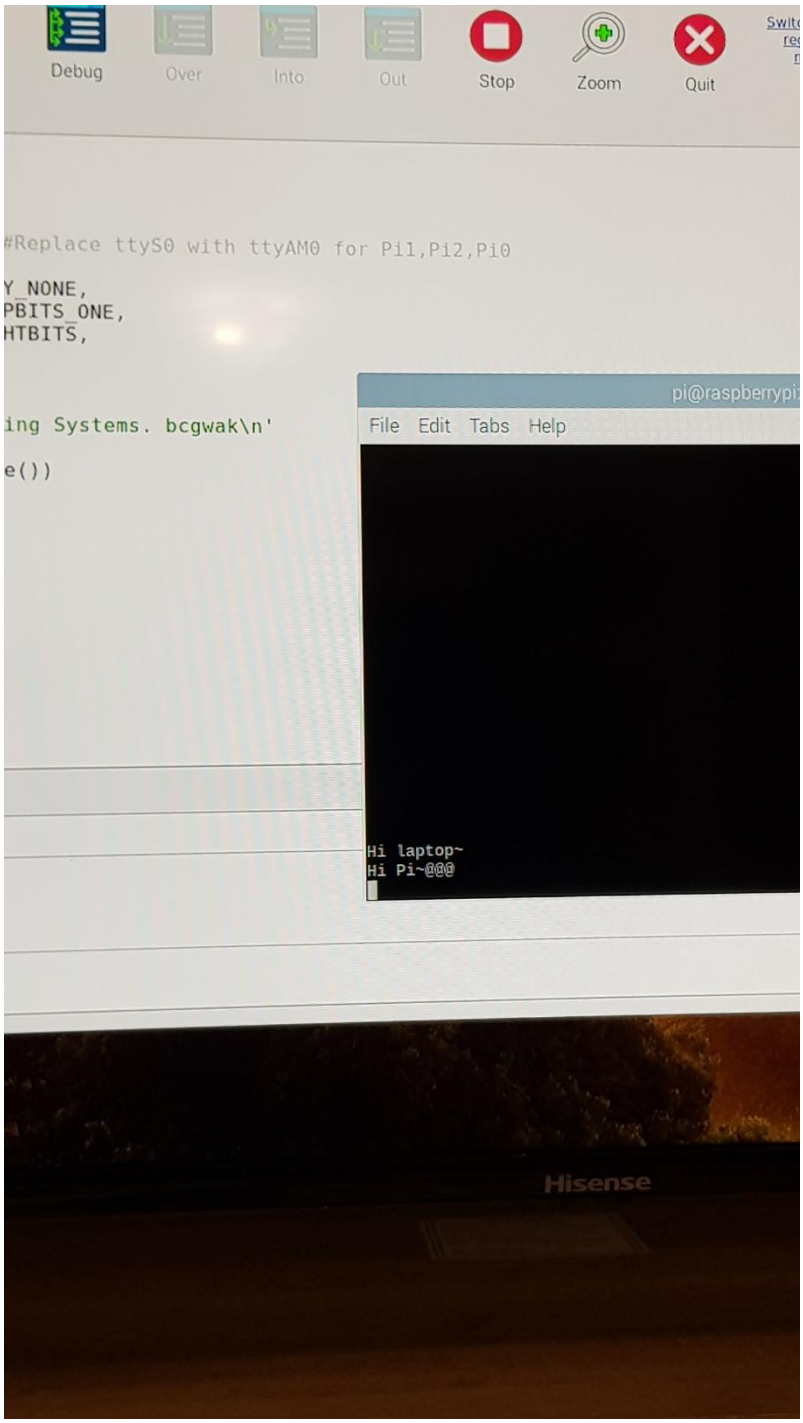
- **Pics for 12.1**

Seding and receiving from my laptop.



Byeongchan Gwak, Student ID : 501026

Seding and receiving from the Pi.

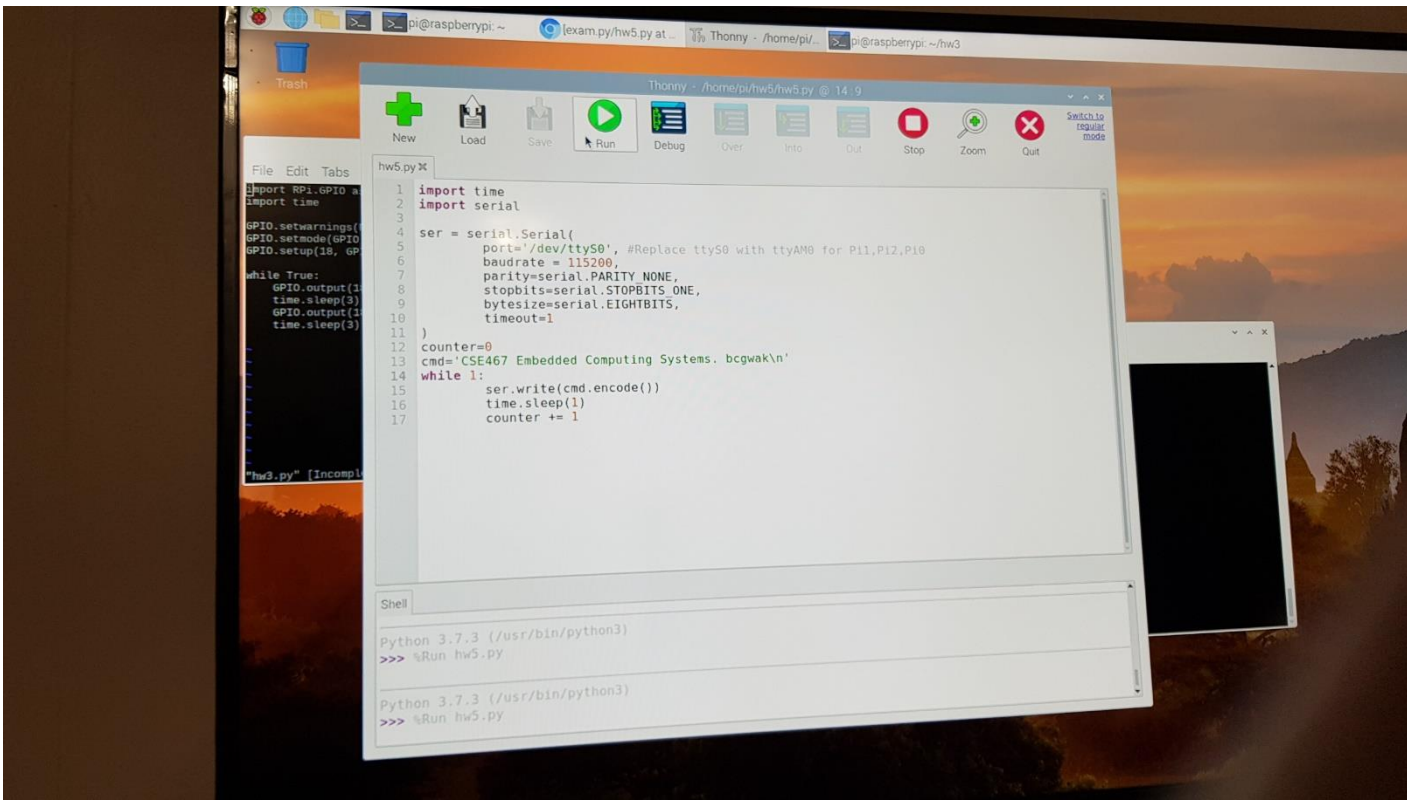


Byeongchan Gwak, Student ID : 501026

- **Pics for 12.2**

Sending the string from Pi to my Laptop using python code.

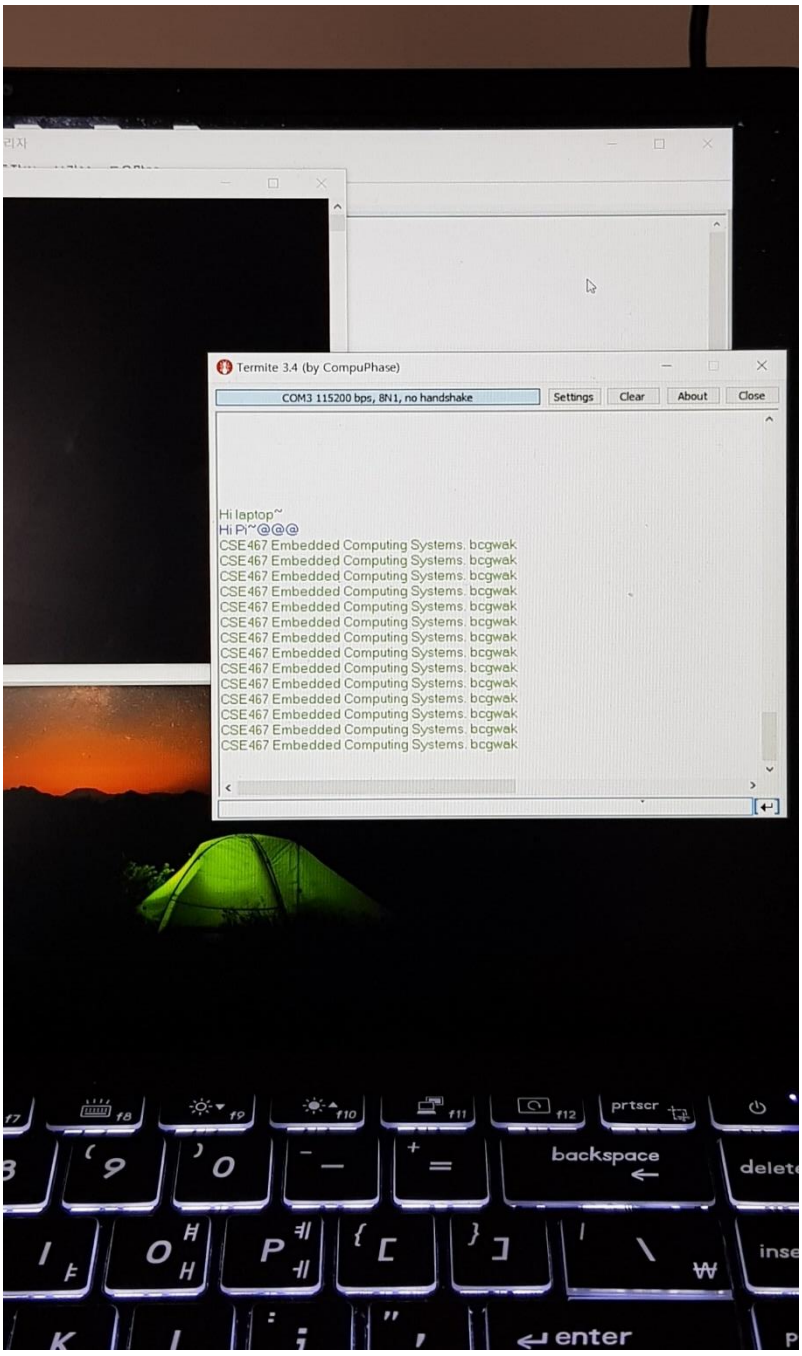
I'm sending 'CSE 467 Embedded Computing Systems. bcgwak' every one second.



Byeongchan Gwak, Student ID : 501026

Receiving the string in my laptop from the Pi.

My laptop receiving endless strings from the Pi



- **12.4**
  - **I attached the video and check it please.**

I really enjoyed the homework!

Thank you so much for your effort, Prof!!