

PROJECT REPORT

Project Title : MASTER-SLAVE communication
through serial port.

Submission on : 13.11.2017



Prepared by
JEMIMA KULANDAISAMY, B.E, M.S(2018)

INDEX:

SL.NO	TOPICS
-------	--------

- | | |
|----|-------------------------------------|
| 1. | TASK DEFINITION. |
| 2. | VERSIONING SYSTEM
WORKFLOW |
| 3. | CODING. |
| 4. | TEST RESULTS. |
| 5. | USAGE DESCRIPTION FOR BOTH
APPS. |

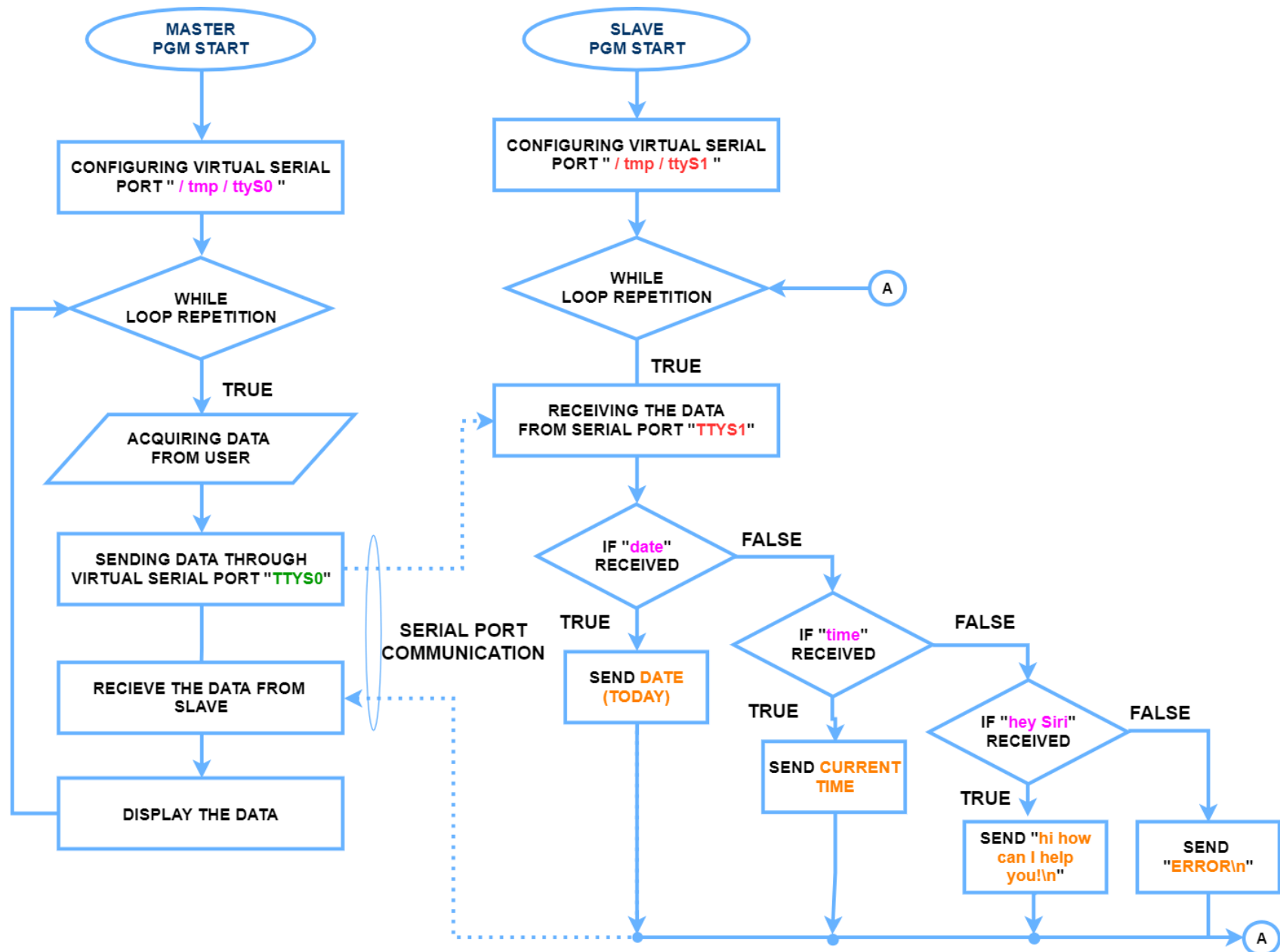
TASK DEFINITION

Write small master-slave applications that communicate through serial port:

CONDITIONS:

1. Use Python 2.x and pySerial library.
2. Use com0com utility to emulate two serial ports <http://com0com.sourceforge.net/>
3. Keep version control flow and use GitHub to store your code.
4. **"Master" app:**
 - a. Will connect to defined Slave application.
 - b. Will allow to send commands in text form.
 - c. Will read responses from slave application.
 - d. Transactions will be human readable.
5. **"Slave" app:**
 - a. Will connect to Master application.
 - b. Will listen to received commands and respond to defined commands.
 - c. On undefined commands will respond with "ERROR" text.
6. Create simple tests that verify the functionality of "Master" app.
7. You will need to design a command protocol upfront
8. You can add some creativity that commands may do something extra in slave application, but it's not mandatory.
9. NO GUI is required, just console. (text based)..

WORKFLOW



CODING

Master program (filename.py)

```
#!/usr/bin/env python2
import serial
import sys

def config():
    ser = serial.Serial("/tmp/ttyS0")
    return ser

def writeToSerial(ser):
    while True:
        input = raw_input("CMD: ")
        ser.write(input + "\n")
        ser.flush()
        print ser.readline()

if __name__ == "__main__":
    ser = config()
    writeToSerial(ser)
```

END OF MASTER PROGRAM (P.T.O)

Slave program (filename.py)

```
#!/usr/bin/env python2

import serial

import sys

import datetime

def config():

    ser = serial.Serial("/tmp/ttyS1")

    return ser

def readFromSerial(ser):

    while True:

        ret = ser.readline().strip("\n")

        cur = datetime.datetime.now()

        if ret == "date":

            val = cur.strftime("%Y-%m-%d")

            ser.write(val + "\n")

        elif ret == "time":

            val = cur.strftime("%H:%M")

            ser.write(val + "\n")

        elif ret == "hey Siri":T

            ser.write("hi how can I help you!\n")

        else:

            ser.write("ERROR\n")

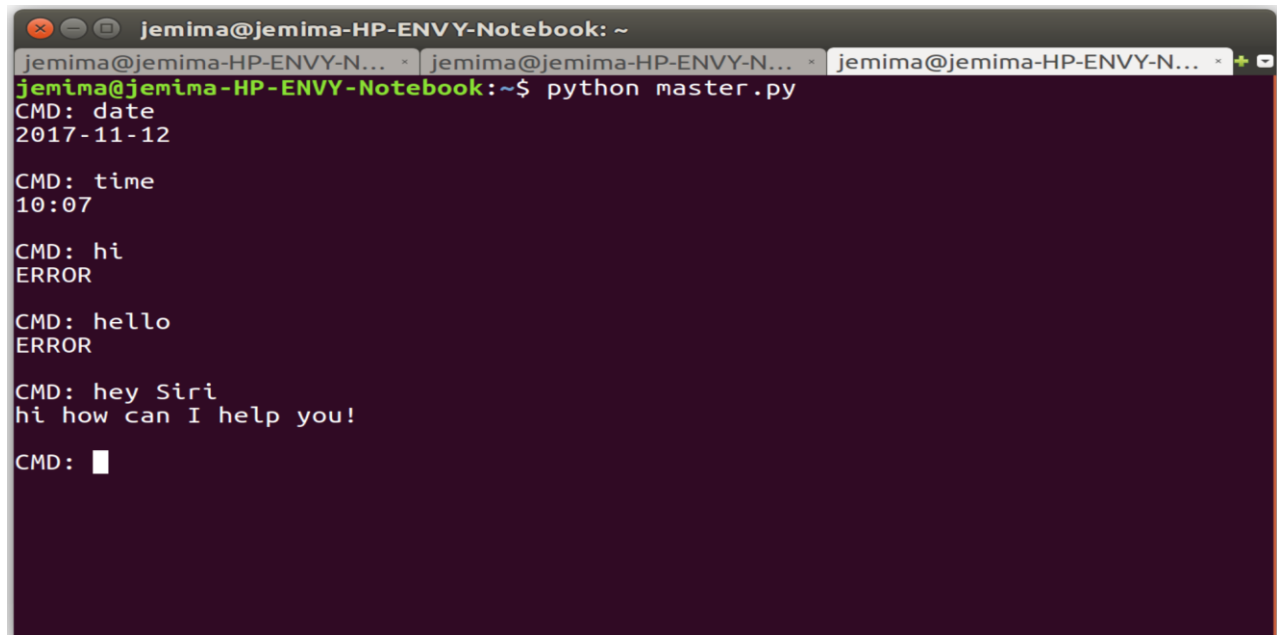
if __name__ == "__main__":

    ser = config()

    readFromSerial(ser)
```

END OF SLAVE PROGRAM

TEST RESULTS



```
jemima@jemima-HP-ENVY-Notebook: ~  
jemima@jemima-HP-ENVY-N... * jemima@jemima-HP-ENVY-N... * jemima@jemima-HP-ENVY-N... *  
jemima@jemima-HP-ENVY-Notebook:~$ python master.py  
CMD: date  
2017-11-12  
  
CMD: time  
10:07  
  
CMD: hi  
ERROR  
  
CMD: hello  
ERROR  
  
CMD: hey Siri  
hi how can I help you!  
  
CMD: █
```

USAGE DESCRIPTION FOR BOTH APPS:

The Master/Slave design pattern is another fundamental architecture python developer's use. It is used when we have two or more processes that need to run simultaneously and continuously but at different rates.

If these processes run in a single loop, severe timing issues can occur. These timing issues occur when one part of the loop takes longer to execute than expected.

If this happens, the remaining sections in the loop get delayed. The Master/Slave pattern consists of multiple parallel loops. Each of the loops may execute tasks at different rates.

Of these parallel loops, one loop acts as the master and the others act as slaves. The master loop controls all of the slave loops, and communicates with them using messaging architectures.

USAGE DESCRIPTION FOR BOTH APPS:

SI.NO	MASTER REQUEST DATA	SLAVE RESPONSE
1	<code>CMD: date</code>	<code>CMD: date 2017-11-12</code>
2	<code>CMD: time</code>	<code>CMD: time 10:07</code>
3	<code>CMD: hey Siri</code>	<code>CMD: hey Siri hi how can I help you!</code>
4	<code>CMD: hello</code>	<code>CMD: hello ERROR</code>

END OF DOCUMENTATION