LABORATORY 5: TIVA – RPI Integration

Part 1: UART.

Theory Concepts:

UART is an asynchronous serial communication protocol that enables the communication between 2 devices. This communication protocol sends 1 byte per time sequentially, for that reason Uart uses different speed for the communication, generally we use 9600 bauds.

Wires	2
Speed	9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1000000, 1500000
Methods of Transmission	Asynchronous
Maximum Number of Masters	1
Maximum Number of Slaves	1

UART uses a packet where a start bit is sent, then the data frame, then the parity bit (if used) and finally the stop bit.

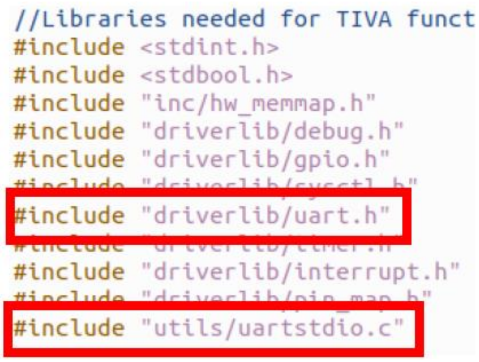
How does the start, parity and stop bit works? Explain it with an example.

UART in TIVA:

To use UART in TIVA we will need to:

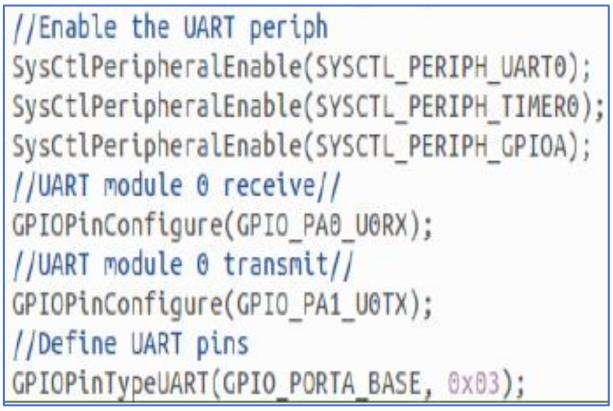
- Import libraries.
- Enable UART ports and peripherals.
- Configure UART baud rate.
- Send and receive data.

Import libraries:



```
//Libraries needed for TIVA funct
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "driverlib/timer.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "utils/uartstdio.c"
```

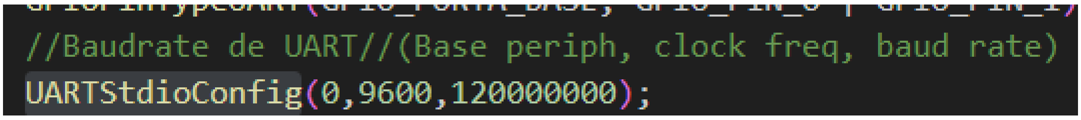
Enable UART:



```
//Enable the UART periph
SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
//UART module 0 receive//
GPIOPinConfigure(GPIO_PA0_U0RX);
//UART module 0 transmit//
GPIOPinConfigure(GPIO_PA1_U0TX);
//Define UART pins
GPIOPinTypeUART(GPIO_PORTA_BASE, 0x03);
```

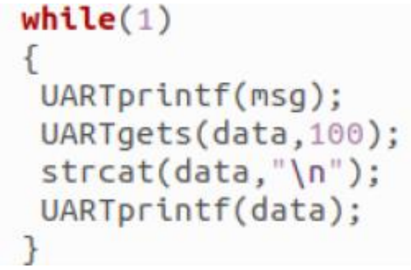
The previous lines go inside the main function. Is necessary to enable the peripheral, configure the pin for transmitter TX and receiver RX and define the pins for UART.

UART Baud Rate:



```
GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
//Baudrate de UART//(Base periph, clock freq, baud rate)
UARTStdioConfig(0,9600,120000000);
```

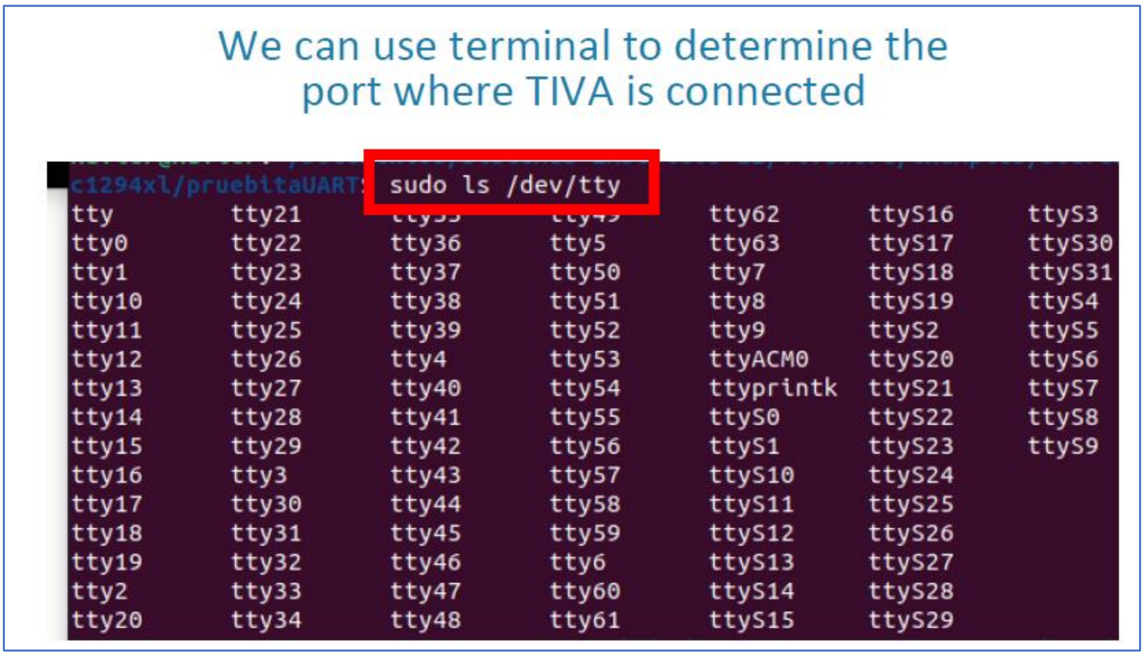
UART use:



```
while(1)
{
 UARTprintf(msg);
 UARTgets(data,100);
 strcat(data,"\n");
 UARTprintf(data);
}
```

UARTprintf() sends data

UARTgets receives data



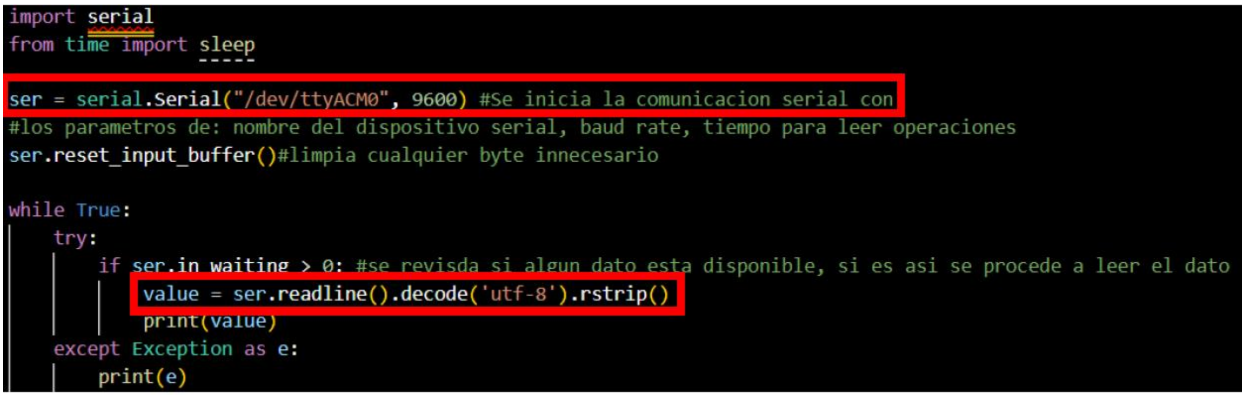
We can use terminal to determine the port where TIVA is connected

```
c1294xl/pruebitaUART$ sudo ls /dev/tty
tty      tty21    tty35    tty49    tty62     ttyS16   ttyS3
tty0     tty22    tty36    tty5     tty63     ttyS17   ttyS30
tty1     tty23    tty37    tty50    tty7      ttyS18   ttyS31
tty10    tty24    tty38    tty51    tty8      ttyS19   ttyS4
tty11    tty25    tty39    tty52    tty9      ttyS2    ttyS5
tty12    tty26    tty4     tty53    ttyACM0   ttyS20   ttyS6
tty13    tty27    tty40    tty54    ttyprintk ttyS21   ttyS7
tty14    tty28    tty41    tty55    ttyS0     ttyS22   ttyS8
tty15    tty29    tty42    tty56    ttyS1     ttyS23   ttyS9
tty16    tty3     tty43    tty57    ttyS10    ttyS24
tty17    tty30    tty44    tty58    ttyS11    ttyS25
tty18    tty31    tty45    tty59    ttyS12    ttyS26
tty19    tty32    tty46    tty6     ttyS13    ttyS27
tty2     tty33    tty47    tty60    ttyS14    ttyS28
tty20    tty34    tty48    tty61    ttyS15    ttyS29
```

UART in Rasp:

For the raspberry, the UART usage is easier than TIVA, we must remember to select the port where our TIVA is connected. Another good practice is to manage all the UART operations using a try/except block.

What is try/except? How do you use it in Python? Which advantages and disadvantages it has?



```
import serial
from time import sleep

ser = serial.Serial("/dev/ttyACM0", 9600) #Se inicia la comunicacion serial con
#los parametros de: nombre del dispositivo serial, baud rate, tiempo para leer operaciones
ser.reset_input_buffer()#limpia cualquier byte innecesario

while True:
    try:
        if ser.in_waiting > 0: #se revisda si algun dato esta disponible, si es asi se procede a leer el dato
            value = ser.readline().decode('utf-8').rstrip()
            print(value)
    except Exception as e:
        print(e)
```

Project

The project will be a small robot, for that Raspberry will activate some motors, to achieve this:

1. Send a message “motor1” or “motor2” using UART when the user switches are pressed.
2. The messages from 1 must be received in the Raspberry and they must activate their respective motors.
3. Both motors must work at 50% of duty cycle.
4. Using SSH modify the motor’s duty cycle. How can we use PWM in raspberry pi? Explain the steps.
5. When a button is pressed on the Raspberry a message “buzzer” must be send to the Tiva.
6. When Tiva receives the message from 5, a buzzer should be activated for 2 seconds.

For the following exercise you will generate a 3-state sequence with TIVA’s user LEDs. This sequence must work using one timer and using SSH you must be able to change the second the timer works. For this you should:

- Create a 3-state sequence in TIVA.
- Implement the sequence in a Timer.
- Using SSH change the seconds the timer takes to generate the sequence.

How can you implement a timer that works only once? Research about one shot timers.