Instituto Tecnológico de Estudios Superiores de Monterrey

Laboratorio Sistemas Embebidos

Práctica 5 - I2C Interfacing with C

Prof. Matias Vázquez Piñón

Gpo 2

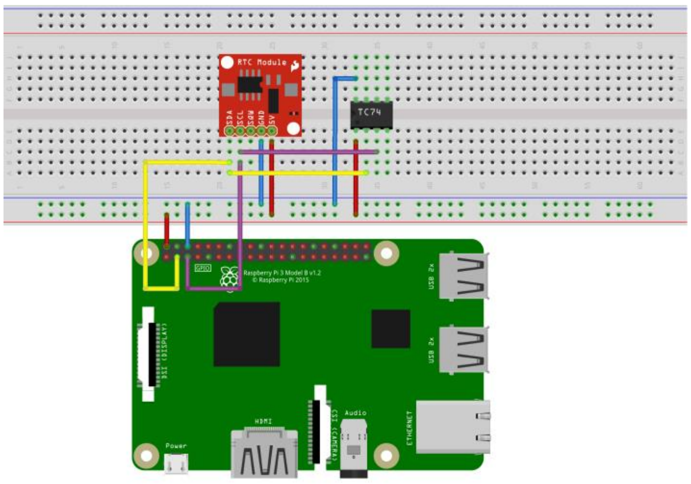
Héctor Javier Pequeño Chairez - A01246364

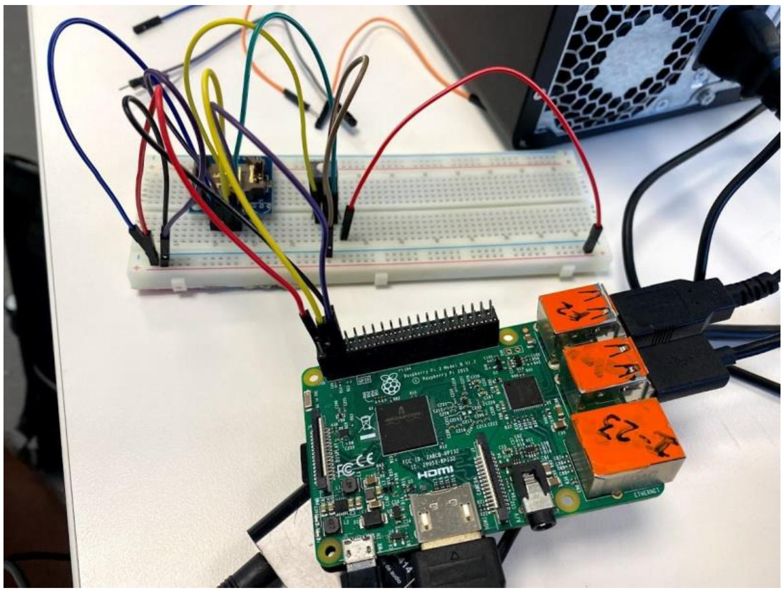
Stephanie Denisse Benítez Cabrera - A00820320

Fecha: 20 / 09 / 2021

Part I: Hardware and Fire Test

The hardware setup, including Raspberry Pi 3 + Tiny RTC I2C module DS1307 + temperature sensor TC74:





In your project, the Raspberry Pi must be able to communicate with both devices through the I2C protocol. Install the Python tools that allow I2C communications between the Raspberry Pi and hooked devices, as well as diagnostic tools:

$ sudo apt-get install -y python-smbus i2c-tools

Once installation is finished, test whether the modules are fully loaded:

*$ lsmod | grep i2c_*

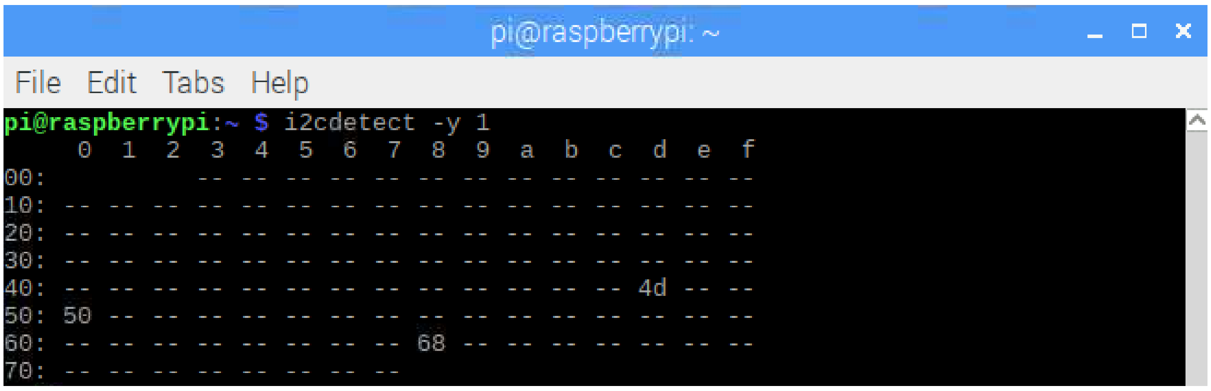
i2c_bcm2835 16384 0

i2c_bcm2708 16384 0

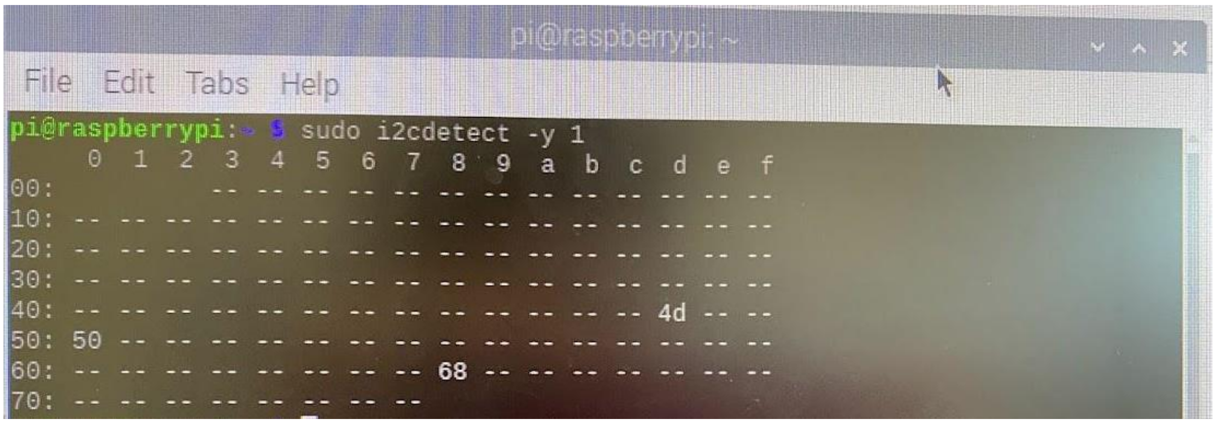
You need the module i2c_bcm2708 (or i2c_bcm2835 on the Raspberry Pi Zero W). If you see the either modules, that means they are correctly loaded. To test the connections and find out the devices' address on the I2C bus, execute:

$ sudo i2cdetect -y 1

If everything goes well, you must see similar to the next figure:



```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- 4d -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```



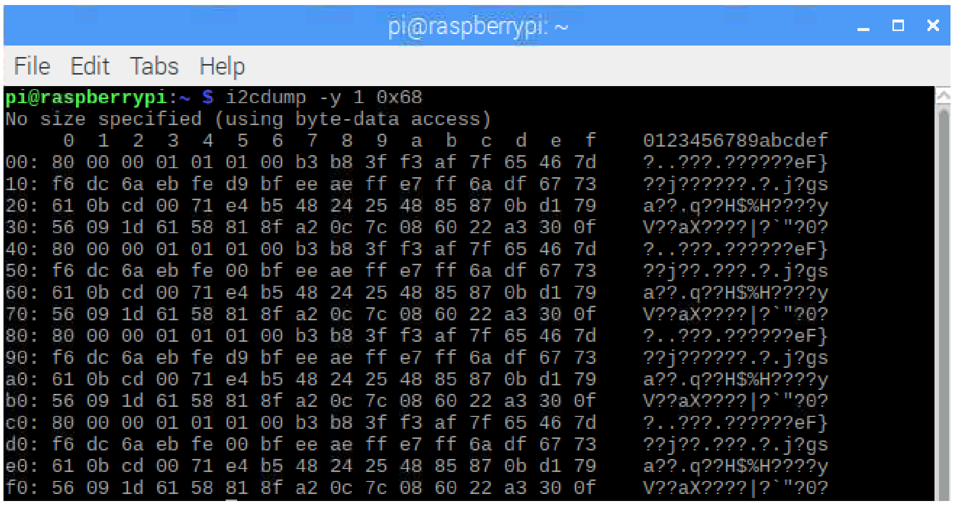
```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- 4d -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Three device' addresses have to be seen in the address map: RPi -> 0x50, TC74 -> 0x4D, and DS1338 -> 0x68.

Examine the register map of a specific device using i2cdump -y 1 [i2c device address]. As an example, to see the register map of the RTC:

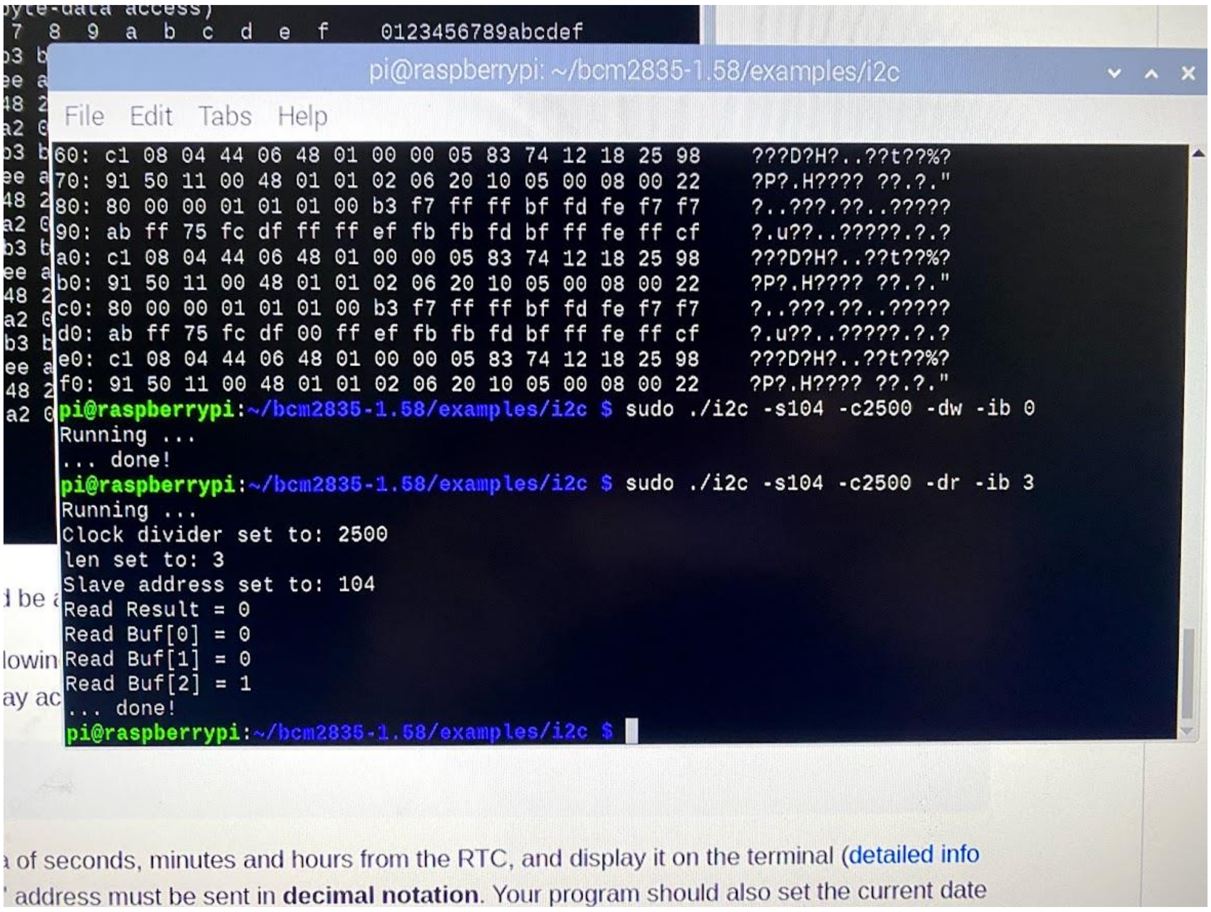
i2cdump -y 1 0x68

Figure below shows the register content for the DS1338 right after been powered on:


pi@raspberrypi: ~

File Edit Tabs Help

```
pi@raspberrypi:~ $ i2cdump -y 1 0x68
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 80 00 00 01 01 01 00 b3 b8 3f f3 af 7f 65 46 7d    ?..???.??????eF}
10: f6 dc 6a eb fe d9 bf ee ae ff e7 ff 6a df 67 73    ??j??????.?.j?gs
20: 61 0b cd 00 71 e4 b5 48 24 25 48 85 87 0b d1 79    a??.q??H$%H????y
30: 56 09 1d 61 58 81 8f a2 0c 7c 08 60 22 a3 30 0f    V??aX????|?`"?0?
40: 80 00 00 01 01 01 00 b3 b8 3f f3 af 7f 65 46 7d    ?..???.??????eF}
50: f6 dc 6a eb fe 00 bf ee ae ff e7 ff 6a df 67 73    ??j??.???.?.j?gs
60: 61 0b cd 00 71 e4 b5 48 24 25 48 85 87 0b d1 79    a??.q??H$%H????y
70: 56 09 1d 61 58 81 8f a2 0c 7c 08 60 22 a3 30 0f    V??aX????|?`"?0?
80: 80 00 00 01 01 01 00 b3 b8 3f f3 af 7f 65 46 7d    ?..???.??????eF}
90: f6 dc 6a eb fe d9 bf ee ae ff e7 ff 6a df 67 73    ??j??????.?.j?gs
a0: 61 0b cd 00 71 e4 b5 48 24 25 48 85 87 0b d1 79    a??.q??H$%H????y
b0: 56 09 1d 61 58 81 8f a2 0c 7c 08 60 22 a3 30 0f    V??aX????|?`"?0?
c0: 80 00 00 01 01 01 00 b3 b8 3f f3 af 7f 65 46 7d    ?..???.??????eF}
d0: f6 dc 6a eb fe 00 bf ee ae ff e7 ff 6a df 67 73    ??j??.???.?.j?gs
e0: 61 0b cd 00 71 e4 b5 48 24 25 48 85 87 0b d1 79    a??.q??H$%H????y
f0: 56 09 1d 61 58 81 8f a2 0c 7c 08 60 22 a3 30 0f    V??aX????|?`"?0?
```


pi@raspberrypi: ~/bcm2835-1.58/examples/i2c

File Edit Tabs Help

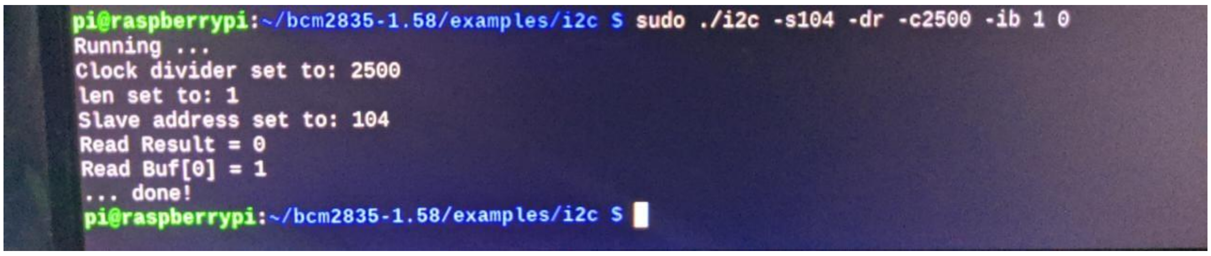
```
60: c1 08 04 44 06 48 01 00 00 05 83 74 12 18 25 98    ???D?H?..??t??%?
70: 91 50 11 00 48 01 01 02 06 20 10 05 00 08 00 22    ?P?.H???? ??.?."
80: 80 00 00 01 01 01 00 b3 f7 ff ff bf fd fe f7 f7    ?..???.??..?????
90: ab ff 75 fc df ff ff ef fb fb fd bf ff fe ff cf    ?.u??..?????.?.?
a0: c1 08 04 44 06 48 01 00 00 05 83 74 12 18 25 98    ???D?H?..??t??%?
b0: 91 50 11 00 48 01 01 02 06 20 10 05 00 08 00 22    ?P?.H???? ??.?."
c0: 80 00 00 01 01 01 00 b3 f7 ff ff bf fd fe f7 f7    ?..???.??..?????
d0: ab ff 75 fc df 00 ff ef fb fb fd bf ff fe ff cf    ?.u??..?????.?.?
e0: c1 08 04 44 06 48 01 00 00 05 83 74 12 18 25 98    ???D?H?..??t??%?
f0: 91 50 11 00 48 01 01 02 06 20 10 05 00 08 00 22    ?P?.H???? ??.?."
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ sudo ./i2c -s104 -c2500 -dw -ib 0
Running ...
... done!
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ sudo ./i2c -s104 -c2500 -dr -ib 3
Running ...
Clock divider set to: 2500
len set to: 3
Slave address set to: 104
Read Result = 0
Read Buf[0] = 0
Read Buf[1] = 0
Read Buf[2] = 1
... done!
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $
```

a of seconds, minutes and hours from the RTC, and display it on the terminal (detailed info
' address must be sent in **decimal notation**. Your program should also set the current date

Now we run an executable program to obtain the data of the seconda, minutes and hours from the RTC, and display it on the terminal.

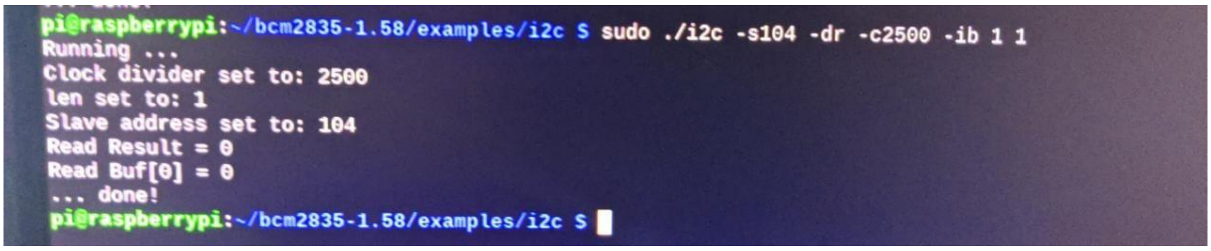
Read the RTC values

Seconds



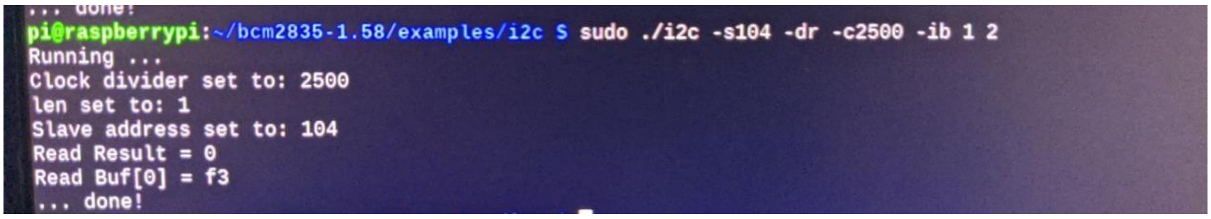
```
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ sudo ./i2c -s104 -dr -c2500 -ib 1 0
Running ...
Clock divider set to: 2500
len set to: 1
Slave address set to: 104
Read Result = 0
Read Buf[0] = 1
... done!
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ 
```

Minutes



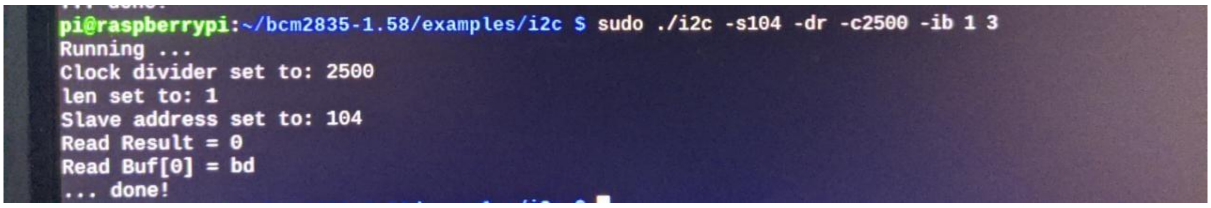
```
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ sudo ./i2c -s104 -dr -c2500 -ib 1 1
Running ...
Clock divider set to: 2500
len set to: 1
Slave address set to: 104
Read Result = 0
Read Buf[0] = 0
... done!
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ 
```

Hour



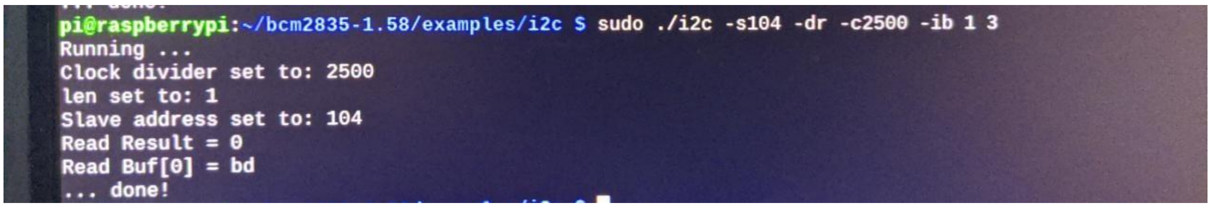
```
... done!
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ sudo ./i2c -s104 -dr -c2500 -ib 1 2
Running ...
Clock divider set to: 2500
len set to: 1
Slave address set to: 104
Read Result = 0
Read Buf[0] = f3
... done!
```

Day Month



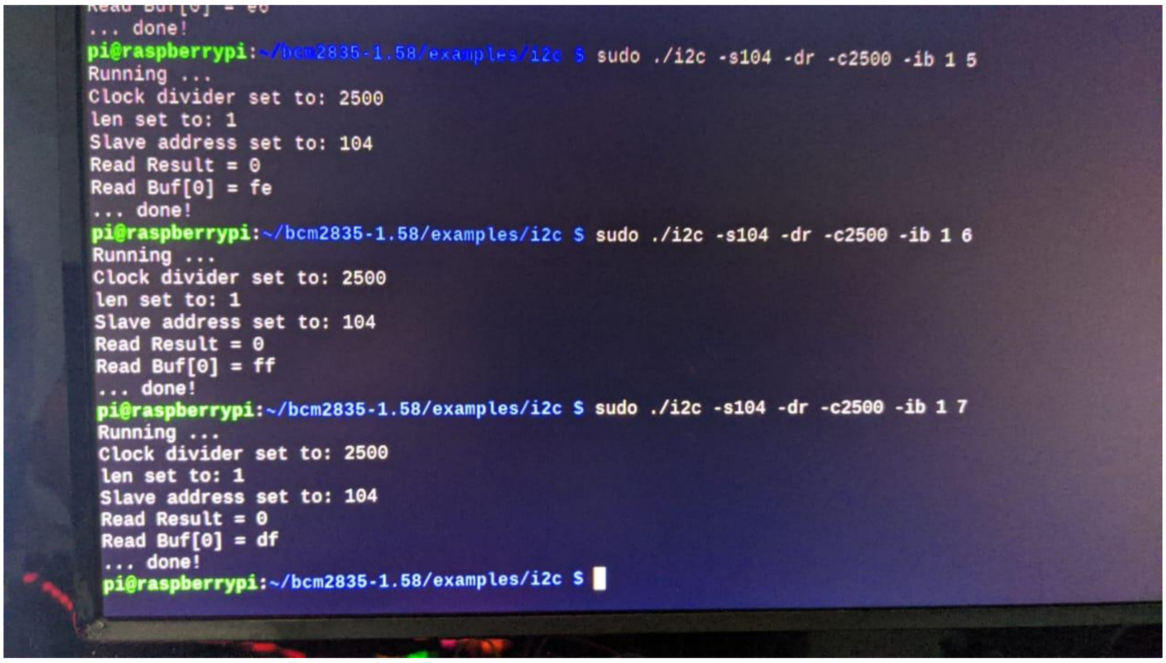
```
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ sudo ./i2c -s104 -dr -c2500 -ib 1 3
Running ...
Clock divider set to: 2500
len set to: 1
Slave address set to: 104
Read Result = 0
Read Buf[0] = bd
... done!
```

Day Week



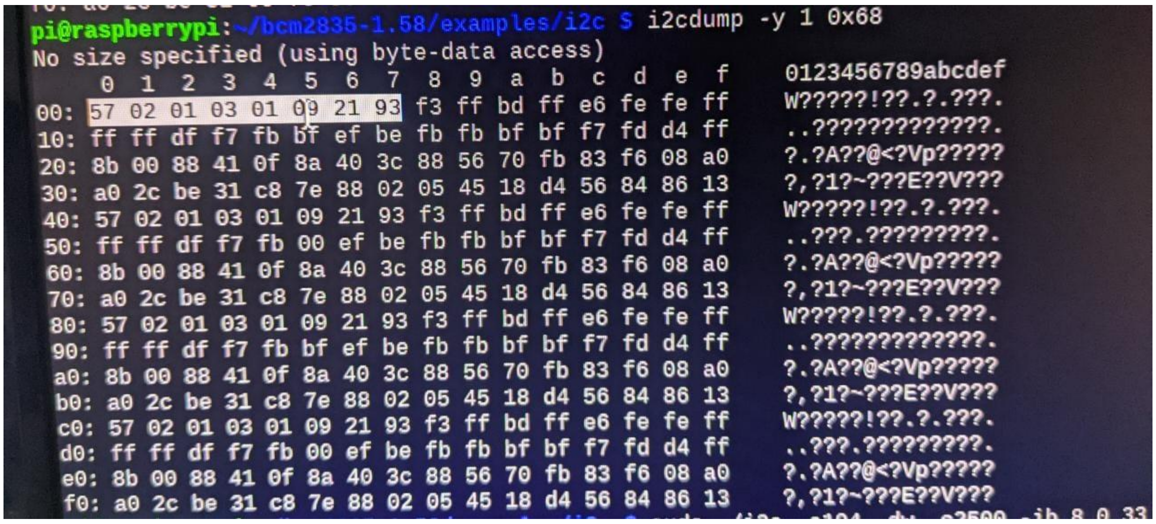
```
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ sudo ./i2c -s104 -dr -c2500 -ib 1 3
Running ...
Clock divider set to: 2500
len set to: 1
Slave address set to: 104
Read Result = 0
Read Buf[0] = bd
... done!
```

Year



```
... done!
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ sudo ./i2c -s104 -dr -c2500 -ib 1 5
Running ...
Clock divider set to: 2500
len set to: 1
Slave address set to: 104
Read Result = 0
Read Buf[0] = fe
... done!
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ sudo ./i2c -s104 -dr -c2500 -ib 1 6
Running ...
Clock divider set to: 2500
len set to: 1
Slave address set to: 104
Read Result = 0
Read Buf[0] = ff
... done!
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ sudo ./i2c -s104 -dr -c2500 -ib 1 7
Running ...
Clock divider set to: 2500
len set to: 1
Slave address set to: 104
Read Result = 0
Read Buf[0] = df
... done!
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $
```

Registers



```
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ i2cdump -y 1 0x68
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 57 02 01 03 01 09 21 93 f3 ff bd ff e6 fe fe ff    W?????!??.?.???.
10: ff ff df f7 fb bf ef be fb fb bf bf f7 fd d4 ff    ..?????????????.
20: 8b 00 88 41 0f 8a 40 3c 88 56 70 fb 83 f6 08 a0    ?.?A??@<?Vp?????
30: a0 2c be 31 c8 7e 88 02 05 45 18 d4 56 84 86 13    ?,?1?~???E??V???
40: 57 02 01 03 01 09 21 93 f3 ff bd ff e6 fe fe ff    W?????!??.?.???.
50: ff ff df f7 fb 00 ef be fb fb bf bf f7 fd d4 ff    ..???.?????????.
60: 8b 00 88 41 0f 8a 40 3c 88 56 70 fb 83 f6 08 a0    ?.?A??@<?Vp?????
70: a0 2c be 31 c8 7e 88 02 05 45 18 d4 56 84 86 13    ?,?1?~???E??V???
80: 57 02 01 03 01 09 21 93 f3 ff bd ff e6 fe fe ff    W?????!??.?.???.
90: ff ff df f7 fb bf ef be fb fb bf bf f7 fd d4 ff    ..?????????????.
a0: 8b 00 88 41 0f 8a 40 3c 88 56 70 fb 83 f6 08 a0    ?.?A??@<?Vp?????
b0: a0 2c be 31 c8 7e 88 02 05 45 18 d4 56 84 86 13    ?,?1?~???E??V???
c0: 57 02 01 03 01 09 21 93 f3 ff bd ff e6 fe fe ff    W?????!??.?.???.
d0: ff ff df f7 fb 00 ef be fb fb bf bf f7 fd d4 ff    ..???.?????????.
e0: 8b 00 88 41 0f 8a 40 3c 88 56 70 fb 83 f6 08 a0    ?.?A??@<?Vp?????
f0: a0 2c be 31 c8 7e 88 02 05 45 18 d4 56 84 86 13    ?,?1?~???E??V???
```

Part II. Bus Topology

1. Based on the i2c.c code, write a program that displays on the terminal and logs in a text file, the current temperature, as read from the TC74, and logs in the date and time every 10 seconds, or each time the temperature exceeds 30°C. Your data must be displayed and logged in the following format shown below.

```
RECEIVER> Temperature: 24°C
RECEIVER> Record 1: 12/09/18 Sat 08:55:44 PM
RECEIVER> Record 2: 12/09/18 Sat 09:01:35 PM
RECEIVER> Record 3: 12/09/18 Sat 09:08:20 PM
```

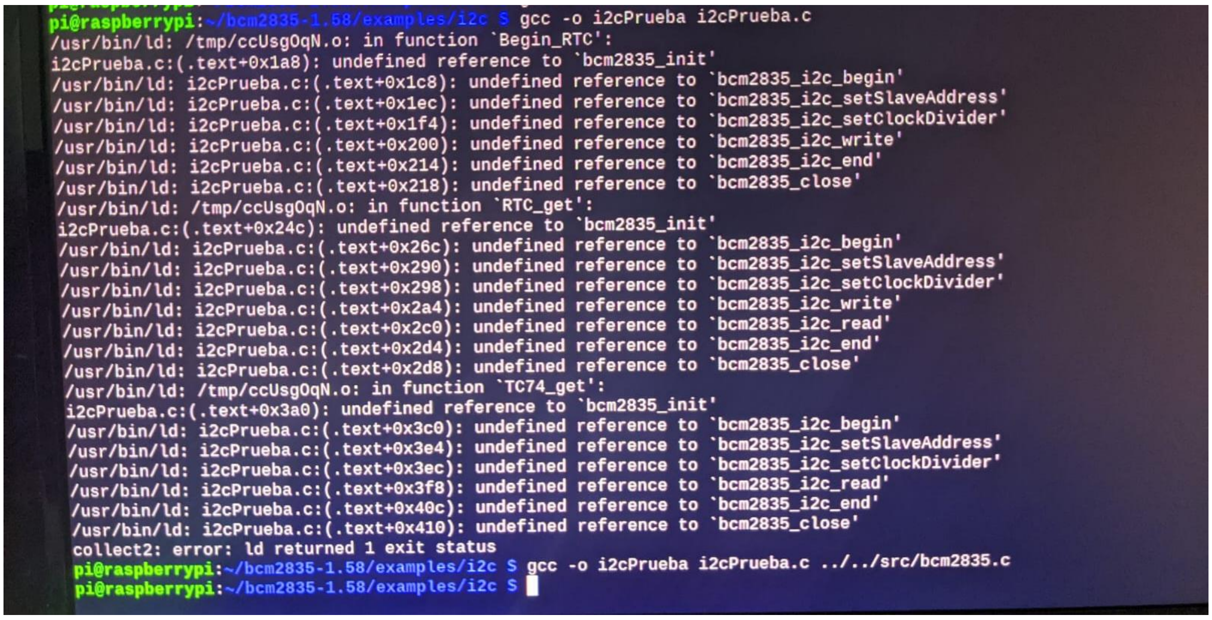
Both, the TC74 and the RTC, should be hooked up to the same I2C bus and must work as slaves; the Raspberry Pi has to be set as master. Consider the following requirements:

- The data log must contain only the 3 most recent events.
- If the data log is full, the newest event replaces the oldest.
- These records must be initialized with the following date:

01/01/01 Mon 12:00:00 AM

Compilacion programa:

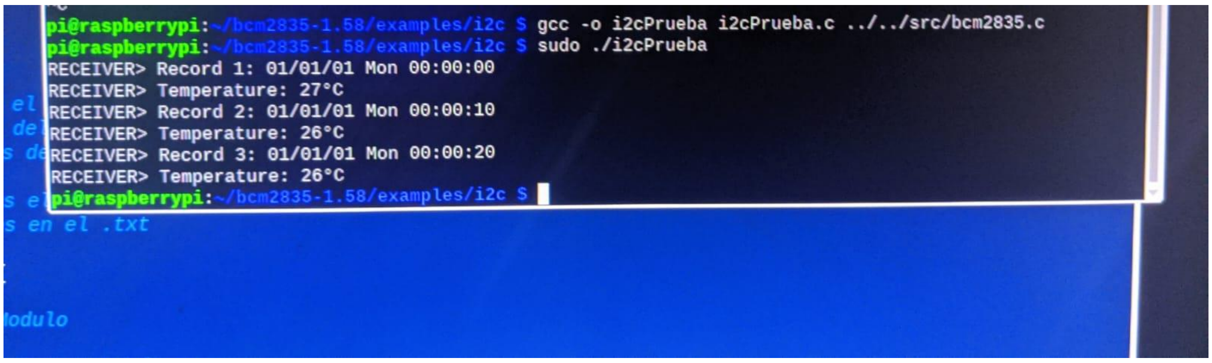
gcc -o i2cPrueba i2cPrueba.c ../../src/bcm2835.c



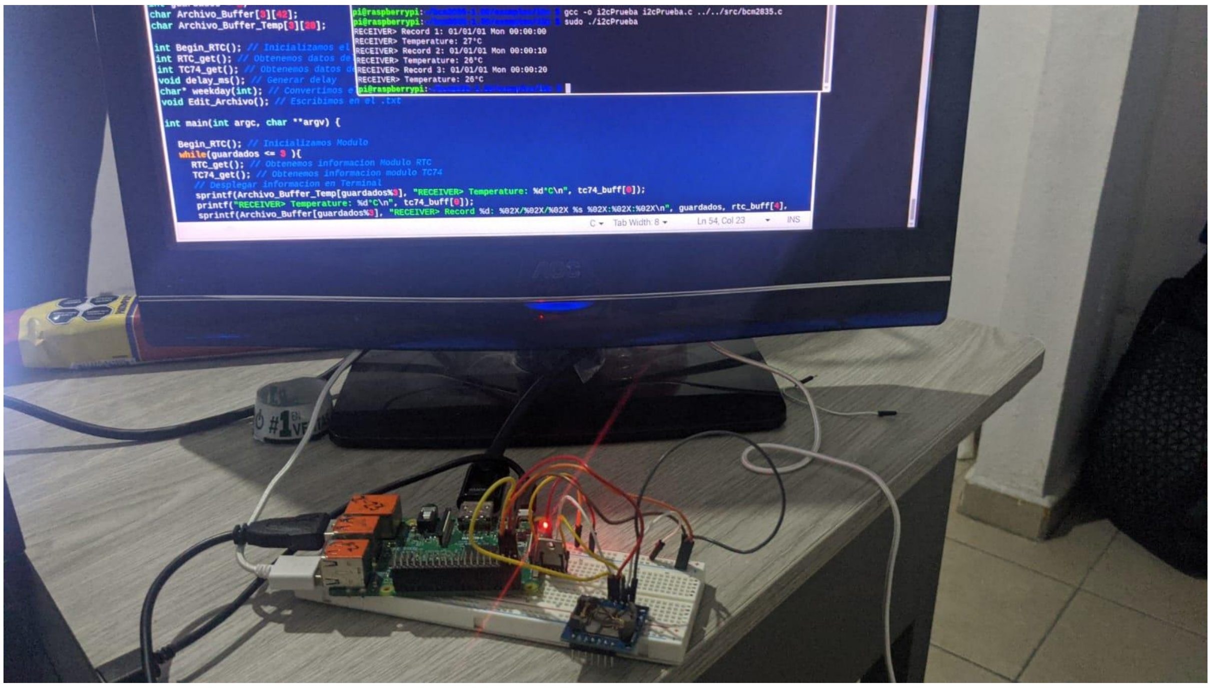
```
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ gcc -o i2cPrueba i2cPrueba.c
/usr/bin/ld: /tmp/ccUsgOqN.o: in function `Begin_RTC':
i2cPrueba.c:(.text+0x1a8): undefined reference to `bcm2835_init'
/usr/bin/ld: i2cPrueba.c:(.text+0x1c8): undefined reference to `bcm2835_i2c_begin'
/usr/bin/ld: i2cPrueba.c:(.text+0x1ec): undefined reference to `bcm2835_i2c_setSlaveAddress'
/usr/bin/ld: i2cPrueba.c:(.text+0x1f4): undefined reference to `bcm2835_i2c_setClockDivider'
/usr/bin/ld: i2cPrueba.c:(.text+0x200): undefined reference to `bcm2835_i2c_write'
/usr/bin/ld: i2cPrueba.c:(.text+0x214): undefined reference to `bcm2835_i2c_end'
/usr/bin/ld: i2cPrueba.c:(.text+0x218): undefined reference to `bcm2835_close'
/usr/bin/ld: /tmp/ccUsgOqN.o: in function `RTC_get':
i2cPrueba.c:(.text+0x24c): undefined reference to `bcm2835_init'
/usr/bin/ld: i2cPrueba.c:(.text+0x26c): undefined reference to `bcm2835_i2c_begin'
/usr/bin/ld: i2cPrueba.c:(.text+0x290): undefined reference to `bcm2835_i2c_setSlaveAddress'
/usr/bin/ld: i2cPrueba.c:(.text+0x298): undefined reference to `bcm2835_i2c_setClockDivider'
/usr/bin/ld: i2cPrueba.c:(.text+0x2a4): undefined reference to `bcm2835_i2c_write'
/usr/bin/ld: i2cPrueba.c:(.text+0x2c0): undefined reference to `bcm2835_i2c_read'
/usr/bin/ld: i2cPrueba.c:(.text+0x2d4): undefined reference to `bcm2835_i2c_end'
/usr/bin/ld: i2cPrueba.c:(.text+0x2d8): undefined reference to `bcm2835_close'
/usr/bin/ld: /tmp/ccUsgOqN.o: in function `TC74_get':
i2cPrueba.c:(.text+0x3a0): undefined reference to `bcm2835_init'
/usr/bin/ld: i2cPrueba.c:(.text+0x3c0): undefined reference to `bcm2835_i2c_begin'
/usr/bin/ld: i2cPrueba.c:(.text+0x3e4): undefined reference to `bcm2835_i2c_setSlaveAddress'
/usr/bin/ld: i2cPrueba.c:(.text+0x3ec): undefined reference to `bcm2835_i2c_setClockDivider'
/usr/bin/ld: i2cPrueba.c:(.text+0x3f8): undefined reference to `bcm2835_i2c_read'
/usr/bin/ld: i2cPrueba.c:(.text+0x40c): undefined reference to `bcm2835_i2c_end'
/usr/bin/ld: i2cPrueba.c:(.text+0x410): undefined reference to `bcm2835_close'
collect2: error: ld returned 1 exit status
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ gcc -o i2cPrueba i2cPrueba.c ../../src/bcm2835.c
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $
```

En este caso apareció un problema debido a que debíamos incluir el archivo bcm2865.c

Ejecución del Programa:



```
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ gcc -o i2cPrueba i2cPrueba.c ../../src/bcm2835.c
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $ sudo ./i2cPrueba
RECEIVER> Record 1: 01/01/01 Mon 00:00:00
RECEIVER> Temperature: 27°C
RECEIVER> Record 2: 01/01/01 Mon 00:00:10
RECEIVER> Temperature: 26°C
RECEIVER> Record 3: 01/01/01 Mon 00:00:20
RECEIVER> Temperature: 26°C
pi@raspberrypi:~/bcm2835-1.58/examples/i2c $
```



```
char Archivo_Buffer[3][42];
char Archivo_Buffer_Temp[3][20];

int Begin_RTC(); // Inicializamos el
int RTC_get(); // Obtenemos datos de
int TC74_get(); // Obtenemos datos d
void delay_ms(); // Generar delay
char* weekday(int); // Convertimos e
void Edit_Archivo(); // Escribimos en el .txt

int main(int argc, char **argv) {

    Begin_RTC(); // Inicializamos Modulo
    while(guardados <= 3 ){
      RTC_get(); // Obtenemos informacion Modulo RTC
      TC74_get(); // Obtenemos informacion modulo TC74
      // Desplegar informacion en Terminal
      sprintf(Archivo_Buffer_Temp[guardados%3], "RECEIVER> Temperature: %d°C\n", tc74_buff[0]);
      printf("RECEIVER> Temperature: %d°C\n", tc74_buff[0]);
      sprintf(Archivo_Buffer[guardados%3], "RECEIVER> Record %d: %02X/%02X/%02X %s %02X:%02X:%02X\n", guardados, rtc_buff[4],
```

```
pi@raspberrypi:[illegible] $ gcc -o i2cPrueba i2cPrueba.c ../../src/bcm2835.c
pi@raspberrypi:[illegible] $ sudo ./i2cPrueba
RECEIVER> Record 1: 01/01/01 Mon 00:00:00
RECEIVER> Temperature: 27°C
RECEIVER> Record 2: 01/01/01 Mon 00:00:10
RECEIVER> Temperature: 26°C
RECEIVER> Record 3: 01/01/01 Mon 00:00:20
RECEIVER> Temperature: 26°C
pi@raspberrypi:[illegible]
```

Conclusion

For this Lab we had to program and develop a program in C using different modules that are RTC and TC74, they both have a real-life functionality. We were able to understand how important these modules are because of their functionality.