Arduino Piggyback on Raspberry Pi

EFY SANI THEO TESTED

SOMNATH BERA

Arduino is a perfect mindless slave. Give it a job and it performs the task endlessly without defaulting even for a nano second. On the other hand, Raspberry Pi, or Raspi, is a computer that has a brain of its own. But, it may falter as it gets busy doing some other job at that moment. Imagine a situation where Arduino is doing the job and Raspberry Pi is supervising it!

Described here is a temperature indicator project using Arduino and Raspi boards. A temperature sensor such as LM35 gives analogue output. Arduino has analogue pins whereas Raspberry Pi does not. LM35 is interfaced directly to Arduino analogue pins and then connected to Raspi. It displays the temperature on the terminal. Let us see how.

Arduino set-up

Arduino development boards are available in many variants in the market. I have used Arduino Uno board for this project. The LM35 temperature sensor is easily available in India. It produces analogue voltage directly proportional to temperature with an output of 1mV per 0.1°C (10mV per degree). The sensor has a temperature range from -55°C to +150°C.

Connect LM35 sensor with Arduino as shown in Fig. 1 and then connect the USB cable to the computer. Open Arduino IDE, compile and upload the following arduino_temp.pde sketch (program) into the UNO board:

```
//arduino-temp.pde
byte n = 0;
const int tpin=0;
void setup(){
  Serial.begin (9600);
  pinMode(tpin, INPUT);
}
void loop()  {
int value=analogRead(tpin);
float mv=(value / 1024.0 )*5000;
float cel = mv / 10;
float far=(cel*9)/5 +32;
delay(1000);
if(Serial.available() ) {
  n = Serial.read();
    Serial.print ("character
    received:");
    Serial.println(char(n));
    Serial.println(n,DEC);
    if(char(n)=='c') {
Serial.print(cel);
Serial.print(":Deg Celsius");
Serial.println("");
    }
    if (char(n)=='f') {
Serial.print(far);
Serial.print( ":Deg Farhenite");
Serial.println("");
    }
    else Serial.print("enter c or f for
    temperature");
    delay(1000);
  }
}
```

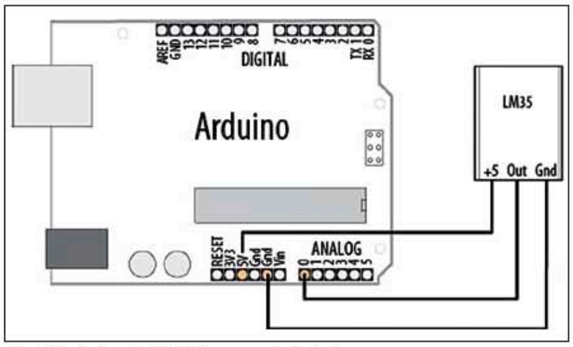
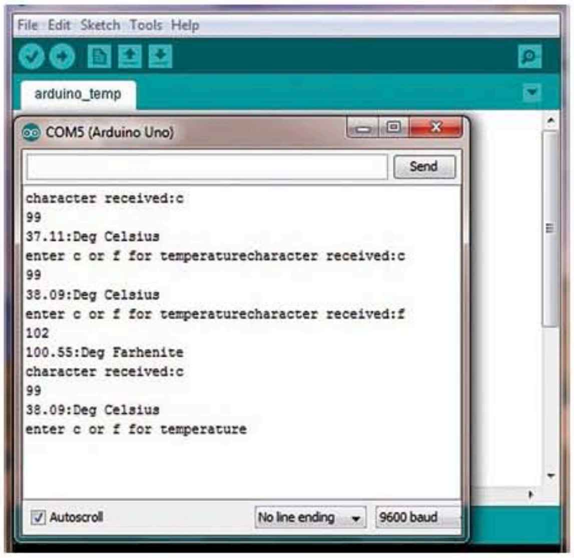


Fig. 1: Interfacing LM35 sensor to Arduino



```
character received:c
99
37.11:Deg Celsius
enter c or f for temperaturecharacter received:c
99
38.09:Deg Celsius
enter c or f for temperaturecharacter received:f
102
100.55:Deg Farhenite
character received:c
99
38.09:Deg Celsius
enter c or f for temperature
```

Fig. 2: Serial monitor

The sketch (program) converts analogRead values into millivolts and divides these by 10 to get degrees.

Run the sketch by opening Serial Monitor in Arduino IDE and write c to get temperature in Celsius and f to get temperature in Fahrenheit (Fig. 2).

So far so good!

Raspberry Pi set-up

You must have a Raspi board with basic set-up and an Internet connection. Install Python, Python-serial and any serial communication software, like Putty or Minicom. I prefer Minicom communication software as it is very light and fast. To install these on Raspi computer, give the following command:

```
$sudo apt-get install
python python-serial
minicom
```

Now, in order to use Raspi's serial port, we need to disable getty (program that displays the login screen). To do this, locate the following line in file /etc/inittab by issuing the command given below:

```
$sudo nano /etc/inittab
```

A screen will pop up, where you need to remove a line that includes getty and 115200. Just comment it out by adding # in front of the line as shown below:

```
#TO:23:respawn:/sbin/getty
-L ttyAMA0 115200 vt100
```

Save it and exit.

Now, when Raspi boots up, it sends data

to the serial ports. We need to stop it from /boot/cmdline.txt file. This is a plain text file, parsed by Raspi as a text file. You need to remove a line from cmdline.text file by issuing the following command. (You need to have a backup file of this, so save the original file as cmdline_backup.txt.)

```
$sudo nano /boot/cmdline.txt
```

Remove console=ttyAMA0, 115200 kgdboc=ttyAMA0, 115220 line.

Save and exit. Reboot your Raspi so that everything comes into effect.

EFY Note

The source codes of this project are included in this month's EFY DVD and are also available for free download at *source.efymag.com*

Arduino-Raspi connection

Arduino pin	Raspi pin
TX	10 (through resistor network as shown in Fig. 3)
RX	Pin 8 (direct)
Gnd	Pin 6 or pin 9 or pin 20 (direct)

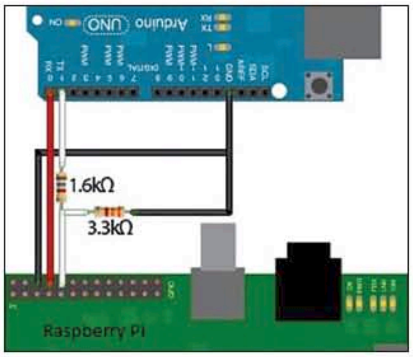


Fig. 3: Connection between Arduino and Raspi

Piggyback configuration

Arduino will sit on top of Raspi and then interact with the LM35 sensor. Arduino will give data to the serial port of Raspi.

GPIO 14 and GPIO 15 (physical pins 8 and 10) of Raspi are UART TX and RX pins. RX and TX pins of Arduino will just go into these GPIO 14 and 15 pins and the ground pin of Arduino will go to any one ground pin (physical pin 6, 9 or 20) of Raspi.

But, do not do that yet. Your Raspi board will be damaged beyond repair. The voltage level of Arduino output pin is approximately 5V, whereas that of Raspi is hardly 3V. We need to reduce Arduino I/O to the matching voltage level of Raspi using the circuit shown in Fig. 3.

Only three wires are to be connected, rest is the power supply that can be separate or common. Serial connection wires, TX and RX, will be interchanged with each other using a resistor network, as shown in the table, on Arduino TX pin to Raspi RX pin (pin 10).

Testing the project

Now, our main intention of making Arduino piggyback on Raspi is on the way. After connecting Arduino and Raspi, as shown in the table and Fig. 3, switch on the power supply of both boards.

Connect Raspi either through a headless mode or in direct mode (I always prefer the headless mode) and then run Minicom as follows:

```
$ ssh -Y pi@192.168.1.4
$ minicom -b 9600 -o -D /dev/ttyAMA0
```

Minicom has a help window that can be accessed by pressing Ctrl+a, z command to see more detail on Minicom.

Minicom will fetch the analogue data from serial port of Raspi and display it on the terminal window. So, by using the command line and Minicom, we can bridge Arduino and Raspi boards for our application. To achieve this, we use Python program temp1.py to access the serial port of Raspi.

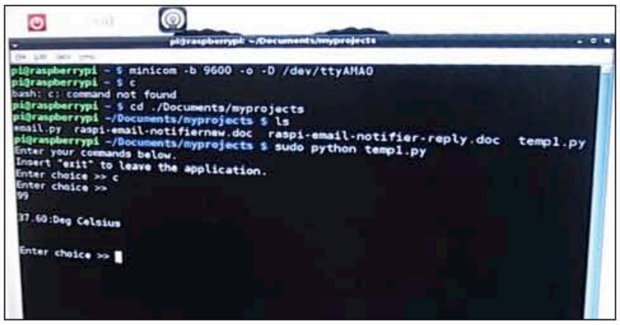
```
# temp1.py
import serial
import time
ser = serial.Serial('/dev/ttyAMA0',9600,
timeout=10)
ser.open()
#ser.write("c")
print 'Enter your commands below.\r\
        nInsert "exit" to leave the
        application.'
input=1
while 1 :
  # get keyboard input
  input = raw_input("Enter choice >> ")
  # Python 3 users
  # input = input(">> ")
  if input == 'exit':
    ser.close()
    exit()
  else:
    ser.write(input)


    out = ''
    time.sleep(1)
    while ser.inWaiting() > 0:
        out += ser.read(1)
    if out != '':
        print "Enter choice >>" + out
```

Run the Python program by issuing following command:

```
$>sudo python temp1.py
```

As the serial console opens in Raspi, enter c to get temperature in Celsius or f to get temperature in Fahrenheit.

This is how you will be able to see temperature data coming from LM35 connected to Arduino, which is sitting piggyback on Raspi. The temperature output reading on the terminal taken during testing is shown in Fig. 4. ●



```
pi@raspberrypi ~ $ minicom -b 9600 -o -D /dev/ttyAMA0
pi@raspberrypi ~ $ c
bash: c: command not found
pi@raspberrypi ~ $ cd ./Documents/myprojects
pi@raspberrypi ~/Documents/myprojects $ ls
email.py  raspi-email-notifiernew.doc  raspi-email-notifier-reply.doc  temp1.py
pi@raspberrypi ~/Documents/myprojects $ sudo python temp1.py
Enter your commands below.
Insert "exit" to leave the application.
Enter choice >> c
Enter choice >>
99

37.60:Deg Celsius

Enter choice >>
```

Fig. 4: Temperature reading on the terminal

Somnath Bera is an avid user of open source software. Professionally, he is a thermal power expert working as additional general manager at NTPC Ltd