USB Arduino Link – Part 3

In previous months I have explained how to set up your Raspberry Pi so that it could communicate with an Arduino via a serial USB cable using Nanpy, how to read digital and analogue values from switches and potentiometers attached to the Arduino’s pins and how to write values to these same pins to control LEDS and a Liquid Crystal Display (LCD); all using Python. This month I will explain how to read temperatures and suggest a few ideas for you to try to increase your skills using what has been covered so far.

**Reading temperature**

A TMP36 is an inexpensive and very easy to use temperature chip which looks like a transistor. If you look at the flat face of the chip, etched with its name, with the legs pointing downwards the left leg goes to 5 volts, the right leg to GND and the centre leg to one of the analogue pins on the Arduino. (See the photograph.) I used A0. Take care to connect it the correct way round after reading the specification at <http://oomlout.com/parts/IC-TMP36-datasheet.pdf> and <http://learn.adafruit.com/tmp36-temperature-sensor> for even more information.

It has a temperature range from -40ºC to 125ºC with a typical accuracy of ±2ºC. It outputs 10mV/ºC with a 500 mV offset to allow for negative temperatures. To use the chip you read the analogue value and convert this to a voltage between 0 and 5V. The program carries out the necessary calculations to convert the voltage to temperatures in ºCelsius.

As it stands the temperature calculated could be up to 2 degrees out in either direction because of the TMP36. There could also be small errors in the analogue to digital converter in the Arduino’s chip. If you need a more accurate reading you can calibrate the system. Connect the TMP 36 with longer wires. Place the sensor in the corner of a polythene bag, about 8-10 cm long, and close the top tightly with an elastic band to keep out the water. Place several ice cubes in a glass with some water and immerse the sensor in the water and melting ice. Wait several minutes until the glass, sensor, water and ice settle to the temperature of melting ice. Run the program again. If the results show a consistent 0ºC, you are in luck. If not, change the value of ‘adjust’ in the script until it gives an average voltage of 0.5 volts while the sensor is in the melting ice and water.

#!/usr/bin/env python

#TMP36 temperature sensor

#Read values from Analogue pin A0

from nanpy import Arduino

from nanpy import serial\_manager

serial\_manager.connect('/dev/ttyACM0')

from time import sleep

tmp36 = 14 # Temp Sensor on A0 - Analog input

adjust = 0 # I needed -0.025 here

print"Reading temperature"

for i in range(0,40):

val = Arduino.analogRead(tmp36) # Analog input

volts = val \* 5.0 /1023 + adjust # Convert to voltage

tempC = volts \* 100 - 50 # Calculate temperature

tempC2 = int(tempC)

print val, volts, tempC, tempC2

sleep(0.3)

To watch the temperature change you can pinch the sensor with warm fingers or add warm water to the glass if the sensor is still in the plastic bag.

The analogue to digital converter has 10 bit accuracy so each bit will represent 5v /1024 = 0.00488 volts. This is equivalent to just under ½ ºC so the set up can only read accurately to the nearest whole degree C.

**Things to try**

Extend the multiple buttons circuit with an extra button and resistor. Then change the program to allow for an extra floor.

If you enjoy soldering you could transfer the five buttons and resistors onto a piece strip board – see photograph. Arrange four of the buttons in a diamond pattern to indicate’ Up’, ‘Down’, ‘Left’ and ‘Right’. Place the fifth button either in the centre of a large diamond or at the side of a small diamond for ‘OK’, ‘Stop’ or’ Fire’. The illustration shows, “one I made earlier”. The copper strips are vertical on the back. There are five horizontal connections. Only one track was cut on the underside. The cut track is under the ‘Up’ button and marked with the magenta blob. This will be useful for the following exercises but the ‘in-line’ version on breadboard will work as well.

Set up three or more LEDs with resistors (330R or more – we do not need them to be too bright) in a row on digital pins.

1. Light the leftmost LED and make the light appear to move from left to right and back again by pressing two buttons - Right & Left. Use Ctrl-C to stop the program.
2. Control the brightness of one LED by two buttons – Up & Down. Use another button to stop the program. NB: the PWM pins are marked with ‘~’: 3, 5, 6, 9, 10 & 11.
3. Combine the first two ideas so that you can set the individual brightness of all the LEDs with 4 buttons. Use the other button to stop the program.

Change the temperature program so that it changes output from ºC to ºF while a button is held down. (Move the output to the LCD if you have one.)

Modify the LCD counting program and control the two sets of numbers with the buttons rather than a counted loop. Use one button pair to increase/decrease the left hand number on the display and another pair to control the right number independently. Stop the program if either number reaches 0.

If you do not have an LCD you could write a similar program but output the two numbers in columns on your monitor – one line each time a button is pressed. (Can you keep the tens and units lined up vertically?) Stop the program with Ctrl-C, but trap the error messages.

There is currently a forum on using the Arduino with Nanpy at:

<http://www.raspberrypi.org/phpBB3/viewtopic.php?p=368522#p368522>

You can use it to ask for help from other users.

Contact me by email via [arduinolink@gmail.com](mailto:arduinolink@gmail.com) with feedback, suggestions or questions.

Thanks again to Andrea Stagi for producing Nanpy.