The Arduino and Me

Just a collection of geeky little projects and tutorials based on the Arduino

Displaying Live Sensor Data in Processing

Introduction

Many a time I have been doing an Arduino project and thought it would be awesome to visualise the data that the various sensors are recording live. I'm aware that some dedicated scripts for both Processing and Arduino exist for this but I found that they were very restrictive and designed for use in isolation. The aim of this tutorial was to show you how to implement your own version which can be integrated into a processing control GUI or your current data transfer protocol between Arduino project and Processing script.

- Rating – Medium
- Total Cost – £0 (As usual the an Arduino and breadboard are not included)

Parts List

- Arduino Uno
- Breadboard
- Jumper Wires
- Sensor of your choice
- USB connection wire

Wiring Schematic

This is entirely up to you and can vary from no more than connecting the arduino to your computer to a fancy sensor setup. I chose to use a HC-SR04 ping sensor because I had a nice board pre-soldered from the Trampoline Bounce Counters (https://thearduinoandme.wordpress.com/projects/trampoline-bounce-counter/) I had made!

Method

The first stage was setting up the Arduino to send a single integer value when requested by the processing script. I prefer to send the values as binary data because it requires much less data transfer than if sent in ASCII format.

```
if(Serial.available()){
int temp = Serial.read();
// *** Replace these lines with your sensor reading code

// I used a ping sensor to measure distance
unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).
int value = uS / US_ROUNDTRIP_CM; // convert from time to distance

// try looking at a floating analog input
//int value = analogRead(A1); // touch the A1 pin and you will see the value change

// *** end of sensor reading code (make sure you store as an integer called value)
Serial.write(value/256);
Serial.write(value%256);
}
```

At this point you should wire up your chosen sensor to your Arduino and test it in isolation. If you are feeling lazy you can simply use a floating analog pin! If you choose to use your own sensor make sure you allocate the readings to an integer called value as I have done. If you were feeling lazy just comment out the ping sensor lines and un-comment the floating analog input line.

The remainder of the method is focused on the processing (https://processing.org/) script running on your PC. The first step is to download the Grafica (https://github.com/jagracar/grafica) library which makes graphing in processing a very simple process. This two way and the instructions on how to do so are here. (https://processing.org/reference/libraries/)

You must then modify the following lines:

points – the number of data points to display at one time
total points – the length of the x axis

```
int points = 450; // number of points to display at a time
int totalPoints = 500; // number of points on x axis
```

Duration – The time in milliseconds between sensor data requests

```
int duration = 20;
```

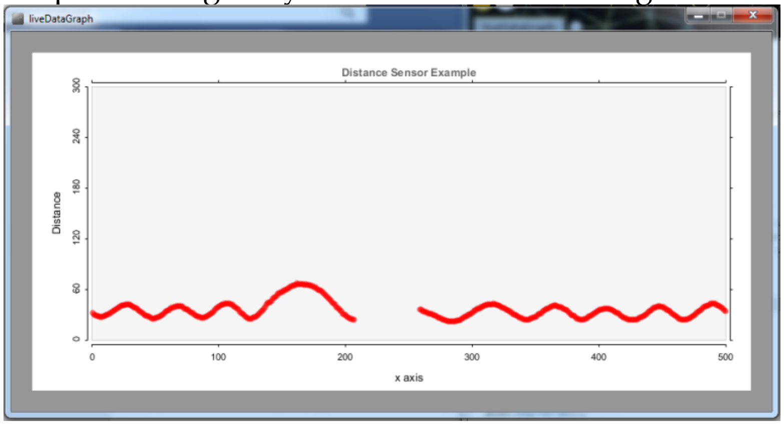
setYLim – Set the limits expected for your sensor (0 – 1024 for standard arduino analog read)

```
plot.setYLim(0, 300); // set y limits
```

Set the titles and labels

```
// Set the plot title and the axis labels
plot.setTitleText("Distance Sensor Example"); // set plot title
plot.getXAxis().setAxisLabelText("x axis"); // set x axis label
plot.getYAxis().setAxisLabelText("Distance"); // set y axis label
```

Once you have done this you are ready to go – Just connect the Arduino up and run the processing script! If all is good you should see something similar to the photo below:



(https://thearduinoandme.files.wordpress.com/2015/06/distance-sensor-example.png)

Arduino Code

```
// ——————————————————————————
// Example NewPing library sketch that does a ping about 20 times per second.
// ——————————————————————————

#include <NewPing.h>

#define TRIGGER_PIN 4 // Arduino pin tied to trigger pin on the ultrasonic sensor.
#define ECHO_PIN 3 // Arduino pin tied to echo pin on the ultrasonic sensor.
#define MAX_DISTANCE 300 // Maximum distance we want to ping for (in centimeters). Maximum
sensor distance is rated at 400-500cm.

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins and
maximum distance.

void setup() {
Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
while(Serial.available()){
int temp = Serial.read();
}

}

void loop() {
if(Serial.available()){
int temp = Serial.read();
// *** Replace these lines with your sensor reading code

// I used a ping sensor to measure distance
unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).
int value = uS / US_ROUNDTRIP_CM; // convert from time to distance

// try looking at a floating analog input
//int value = analogRead(A1); // touch the A1 pin and you will see the value change

// *** end of sensor reading code (make sure you store as an integer called value)
Serial.write(value/256);
Serial.write(value%256);
}

}
```

Processing Code

```
// import require libraries
import grafica.*;
import processing.serial.*;

// create plot instance
GPlot plot;

// initialise global variables
int i = 0; // variable that changes for point calculation
int points = 450; // number of points to display at a time
int totalPoints = 500; // number of points on x axis
float noise = 0.1; // added noise
float period = 0.35;
long previousMillis = 0;
int duration = 20;

// Serial
Serial myPort; // Create object from Serial class

void setup(){
// set size of the window
size (900,450);

// set up serial connection
String portName = Serial.list()[0];
myPort = new Serial(this, portName, 115200);

// initialise graph points object
GPointsArray points1 = new GPointsArray(points);

// calculate initial display points
for (i = 0; i < points; i++) {
points1.add(i,0);
}

// Create the plot
plot = new GPlot(this);
plot.setPos(25, 25); // set the position of to left corner of plot
plot.setDim(750, 300); // set plot size

// Set the plot limits (this will fix them)
plot.setXLim(0, totalPoints); // set x limits
plot.setYLim(0, 300); // set y limits

// Set the plot title and the axis labels
plot.setTitleText("Distance Sensor Example"); // set plot title
plot.getXAxis().setAxisLabelText("x axis"); // set x axis label
plot.getYAxis().setAxisLabelText("Distance"); // set y axis label

// Add the two set of points to the plot
plot.setPoints(points1);

}
```

```
void draw() {
// set window background
background(150);

// draw the plot
plot.beginDraw();
plot.drawBackground();
plot.drawBox();
plot.drawXAxis();
plot.drawYAxis();
plot.drawTopAxis();
plot.drawRightAxis();
plot.drawTitle();
plot.getMainLayer().drawPoints();
plot.endDraw();

// check if i has exceeded the plot size
if (i > totalPoints){
i=0; // reset to zero if it has
}

// get new value from serial port
if ( millis() > previousMillis + duration) { // If data is available,
myPort.write(0);
println("Sensor request sent");
while (myPort.available() < 0){};
int val = myPort.read()*256 + myPort.read(); // read it and store it in val
println(val);
// Add the point at the end of the array
i++;
plot.addPoint(i,val);

// Remove the first point
plot.removePoint(0);

previousMillis = previousMillis + duration;

}

}
```

One thought on "Displaying Live Sensor Data in Processing"

1. **Displaying Live Data | The Arduino and Me** says:
   JUNE 24, 2015 AT 9:22 PM
   […] Displaying Live Sensor Data in Processing […]

REPLY