Lab 6: Fun with Arduino #1: Process Control

Introduction:

There are two main systems of control: Feedback and Feedforward control.

Figure 1: Feedforward Control block diagram

Figure 2: Feedback Control Block Diagram

Feedforward control systems are systems that acts only when there is a deviation in the output from the equilibrium before it affects the system. An external sensor directly communicates with the plant (the one with the transfer function *Gp* in the Figures above). Thus decreasing the ability for the system to deviate from equilibrium. The block diagram for feedforward control is shown in Figure 1.

Feedback control systems are systems that make use of the output which is then fed back into the input to produce an error signal that drives the system. It only acts when the system is already not at equilibrium.

Figure 2 shows the basic block diagram for feedback control systems. *Gc* is the transfer function of the controller. *Hs* represents the sensor which reads the output that is fed back into the input. Ideally the sensor has a value of 1 so that the units of the output is the same as the inputs.

There are two types of feedback controls: Positive and Negative feedback. Positive feedback increase the error signal (inputs and outputs are added together) and thus it tends to move systems away from its equilibrium point making the system unstable. Negative feedback tends to dampen the error signal (inputs and outputs are subtracted from each other). This increases the stability of a system as it tends to keep the system in an equilibrium state.

Bang-Bang controllers (also known as ‘ON-OFF’ controllers) are the simplest type of feedback control system. When the output reaches a certain level, the input is switched off. These are often used to control small home appliances such as small heaters.

The aim of this lab is to implement a ‘Bang-Bang’ controller using Arduino. In this lab, a small lamp will be used as a small heater that has to be controlled when the temperature has fallen below a set ‘LOW’ temperature (i.e. when the temperature gets colder than what was wanted). The lamp should turn off when the temperature reaches a set ‘HIGH’ temperature.

The negative feedback system block diagram for this lab is shown below:

Figure 3: Feedback Control
block diagram for the lab

The error signal for this lab can be characterised as $e = t_{setpoint} - t_{out}$. The set temperature that the system for this lab should be centred around is $26^{o}C$.

Experiment:
Voltage Relay
Equipment and Set up:

- Seeeduino and cable
- Voltage relay box and IEC power cable
- Lamp
- wires

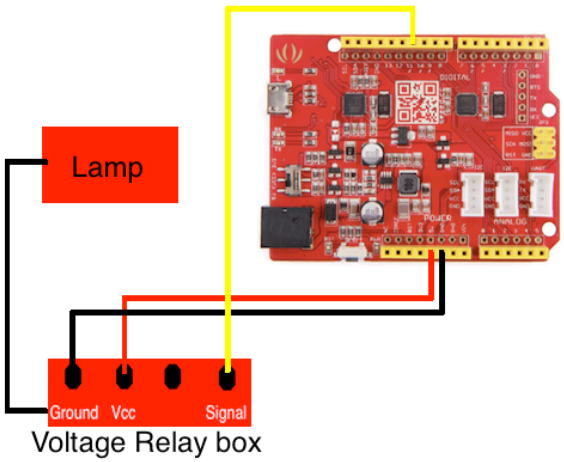

Figure 4: Voltage Relay and Arduino set up

Method:

1. Attach the lamp and the voltage relay to the Arduino. Connect the signal to pin 11, Ground to Ground and Vcc to 5V (see Figure 4).
2. Use the BLINK sketch, from the examples in the Arduino, to make the lamp turn on and off periodically[1].

Results:
The command *delay(x)* pauses the program for the amount of time specified as a parameter, x in milliseconds. (NB. 1000 milliseconds = 1 second)

The fastest time that the lamp could be observed as turning on and off is around .025 seconds (or 25 milliseconds).

[1] or see '*Blink.ino*' in the back

Temperature sensor
Equipment and Set up:

- Seeeduino and cable
- Grove Temperature sensor
- wires

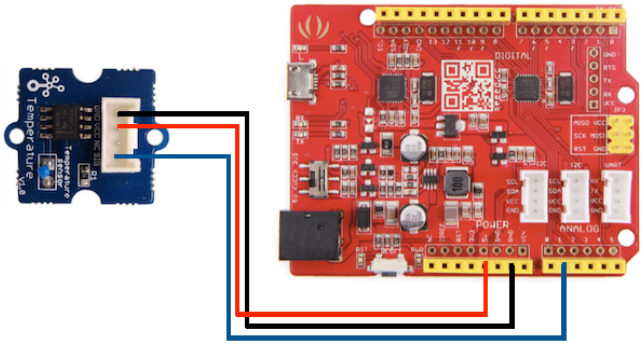

Figure 5: Arduino and temperature sensor set up

Method:

1. Attach the temperature sensor to the Arduino. Connect Vcc to 5V, Ground to Ground of the ELVIS and signal to Analog 1. See Figure 5.
2. Go to the Seeeduino website[2], copy and upload the code for the temperature sensor to the Arduino.
3. Edit the code so that a suitable temperature is printed in the serial monitor.

Results:

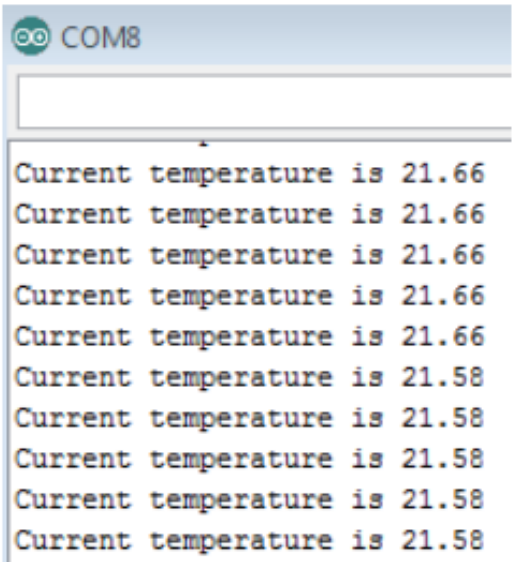

Figure 6: Temperature output from the serial monitor

[2] http://wiki.seeedstudio.com/Grove-Temperature_Sensor/ or see '*tempSensor.ino*' in the back

Figure 6 shows the result of the code from the serial monitor at 9600 bits/sec. The results are updated every 1 second using a *delay()* function.

Temperature Sensor

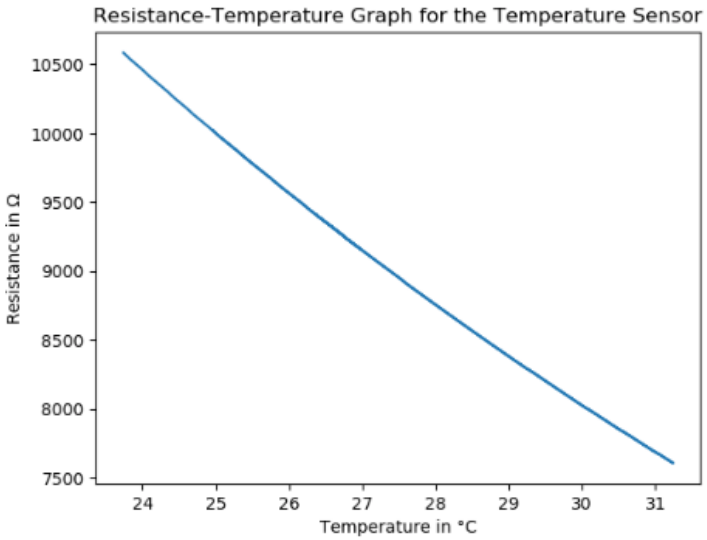

Figure 7: Resistance-Temperature graph

The temperature sensor uses a thermistor to detect the surrounding temperature. The thermistor has a resistance that becomes smaller as the temperature goes higher (hotter). This can be seen on Figure 7. A reading of temperature and resistance is saved and produced this graph.

Analog Read Function

A	B	C
a	r	t
511.00	10,019.57	24.96
511.00	10,019.57	24.96
512.00	9,980.47	25.04
512.00	9,980.47	25.04
513.00	9,941.52	25.13
513.00	9,941.52	25.13
513.00	9,941.52	25.13
513.00	9,941.52	25.13
514.00	9,902.72	25.22
515.00	9,864.08	25.31
517.00	9,787.23	25.48
518.00	9,749.04	25.57
519.00	9,710.98	25.66
520.00	9,673.08	25.75
521.00	9,635.32	25.83
522.00	9,597.70	25.92
524.00	9,522.90	26.10
525.00	9,485.71	26.19

Figure 8: Serial monitor output put on a spreadsheet

The Analog read function, even though it is of a ‘float’ type, only prints out results to the closest integer as shown on Figure 8 under the heading ‘a’.

The analog pin when it is not in use (or not plugged in) returns the number ‘1023.00’. In this setup, the analog pins map inputs to voltages between 0 (ground) and 5V (signal input) into integer values from 0 to 1023. Thus the smallest voltage difference the analog read function can differentiate is 5V/1024= 0.0049V (or 4.9mV).

User Interface
Equipment and Set up:

- Seeeduino and cable
- LCD display shield

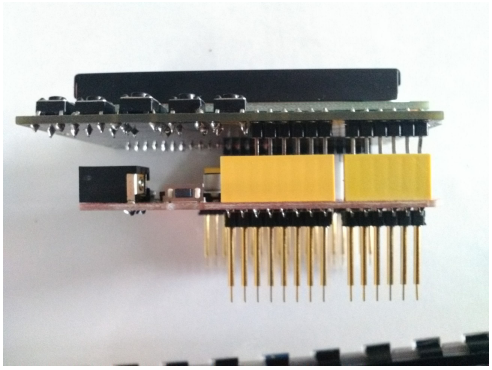

Figure 9: Button side set up

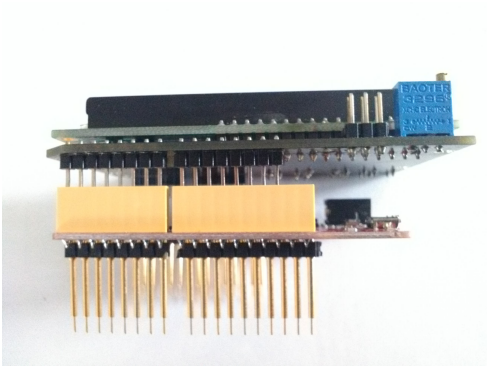

Figure 10: LCD side set up

Pin	Function
Analog 0	Button (select, up, right, down and left)
Digital 4	DB4
Digital 5	DB5
Digital 6	DB6
Digital 7	DB7
Digital 8	RS (Data or Signal Display Selection)
Digital 9	Enable
Digital 10	Backlit Control

Figure 11: Pin allocation for the LCD

Method:

1. Add the LCD display shield to the Arduino and temperature sensor (consult Figure 11 for pin allocation). Figures 9 and 10 shows the attachment of the LCD to the Arduino.
2. Go to the DFROBOT website[3], copy and upload the code for the LCD.
3. Edit the code such that it shows the temperature read from the temperature sensor.

[3]https://www.dfrobot.com/wiki/index.php/Arduino_LCD_KeyPad_Shield_(SKU:_DFR0009) or see '*lcd.ino*' in the back

Results:

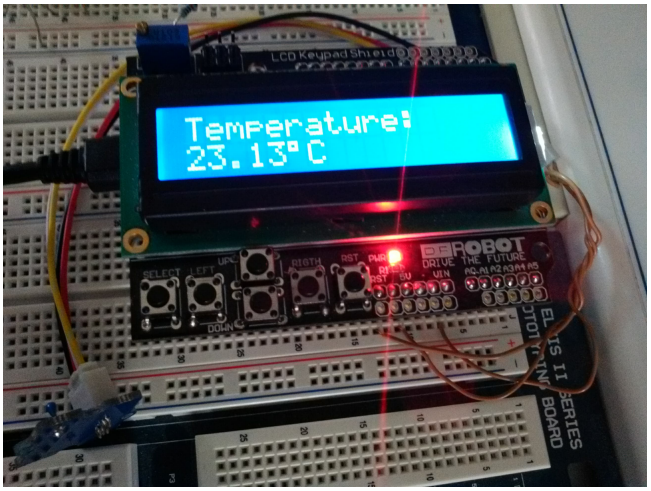

Figure 12: LCD output

Figure 12 shows the temperature as read from the temperature sensor in degree Celsius.

There are two lines on the lcd in which text can be printed on. The most characters that can be printed in one line on the lcd is 16 characters. The cursor has to be set using an *lcd.setCursor(x, y)* function to be able to print on to different parts on the lcd. The (x, y) represents the vector positions on the lcd screen. The default allocated cursor is (0,0) which is the left top side of the lcd. The furthest cursor allocation is (16, 2) which is the bottom right side.

ON/OFF Control System
Equipment and Set up:
Implement the Arduino, voltage relay, temperature sensor and LCD to create a “Bang-bang” controller to act as a thermostat.

Method:

1. Add on the voltage relay and the lamp to the Arduino with the LCD display shield and the temperature sensor (see Figure 4 for the voltage relay connection).
2. Edit the code such that the lamp turns on when the temperature drops to a ‘LOW’ temperature and turns off when the temperature reaches a ‘HIGH’ temperature.
3. Edit the code to show the current temperature and target temperatures on the lcd.
4. Edit the code and make use of the buttons on the lcd display shield to control the target temperatures.

Results:

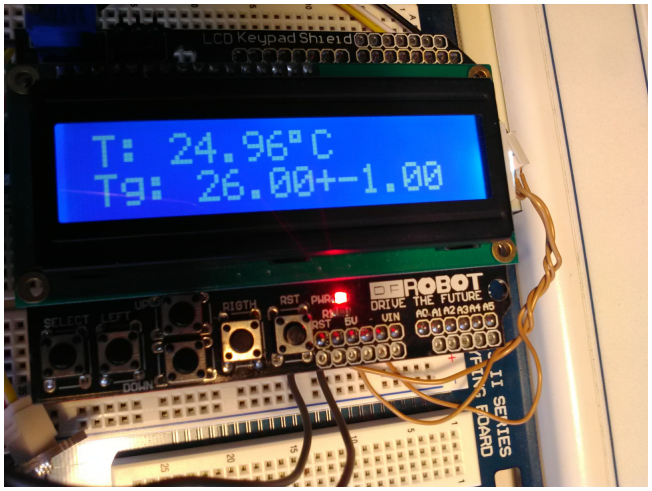

Figure 13: Thermistor output

Figure 13 shows the result of the code[4]. ’*T*’ shows the current temperature. ‘*Tg*’ shows the target temperatures and the standard deviation.

As shown on Figure 13 by the light on the bottom edge, the lamp turns on when the temperature drops to below 25 degrees which is the ‘LOW’ temperature in this set. At temperatures above 27 degrees, the lamp should turn off.

Button Control
The buttons when pressed produces the following results:

- select = 740
- left = 503
- up = 142
- down = 327
- right = 0

[4] Full code ‘*On-off.ino’ in the back*

To be certain of these button outputs and account for a possibly changing values (especially if the lcd display shield is changed), the numbers set to distinguish these buttons are:

- right < 50
- up < 250
- down < 450
- left < 650
- select < 850

The ‘up’ and ‘down’ buttons change the target temperature, Tg, by 1 degree (‘up’ to increase and ‘down’ to decrease). The ‘left’ and ‘right’ buttons adjusts the deviation temperature on the target (this sets the ‘HIGH’ and ‘LOW’ temperatures) by 1 degree (‘right’ to increase and ‘left’ to decrease). The increment (or decrement) rate can be changed easily in the code.

Each time a button is pressed, it sometimes produces some noise (i.e. it sometimes say that a button is pressed a few times when in reality it was only pressed once). This is called debounce. To avoid this, a *delay()* function was added to each button. The response of the button will only happen after a certain period has passed therefore ignoring the noise after the button is pressed.

A better way to avoid debouncing would be to make use of the time setting in the Arduino. Save the last button state and check if it has changed after a certain period with the Arduino clock. Thus the use of delays for each button pressed can be avoided and the button state is more accurate and stable.

Temperature and Time

Figure 14: Temperature-Time relationship

Figure 14 shows the expected relationship between the temperature and the time period. It shows a sinusoidal wave that is centred around the set point of 26^oC. This is expected as the temperature will continue to rise until it hits the ‘HIGH’ set point and the lamp turns off and thus the temperature will go down until it hits the ‘LOW’ set point at which the lamp turns back on to increase the temperature (make it hotter).