

Soldering Iron Safety with the Portlet

Or basically using the Portlet as a timed outlet

Summary: This is a quick example to show how the Portlet can be used as a safety device to keep you from accidentally leaving your soldering iron on. This same concept of putting a timer on an outlet can be used in a lot of applications, but since you are interested in the kit, you are probably interested in Soldering Irons. The code for programming the Arduino is pasted in Appendix A

Table of Contents

Overview	2
Programming the Portlet.....	2
The Setup.....	5
Defining the outlet operation.....	6
Appendix A: Sample Arduino Code.....	7

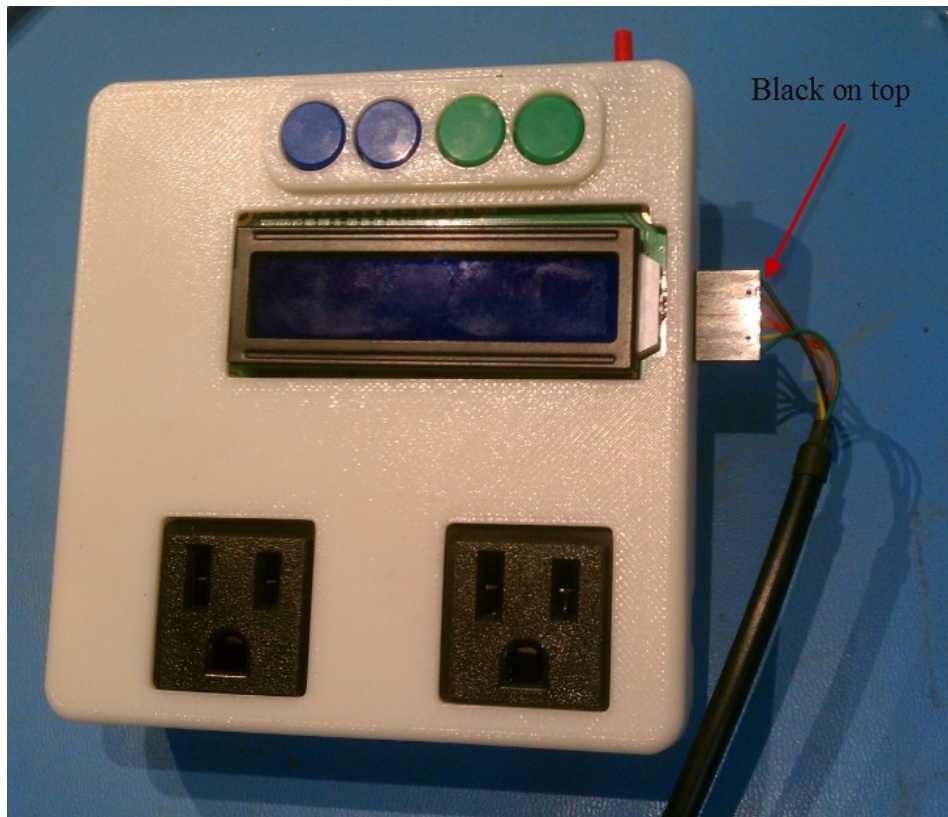
Overview

The concept is very simple: You don't want to leave your soldering iron accidentally. So we program the Portlet to only leave the iron on for an hour at a time before turning off the outlet.

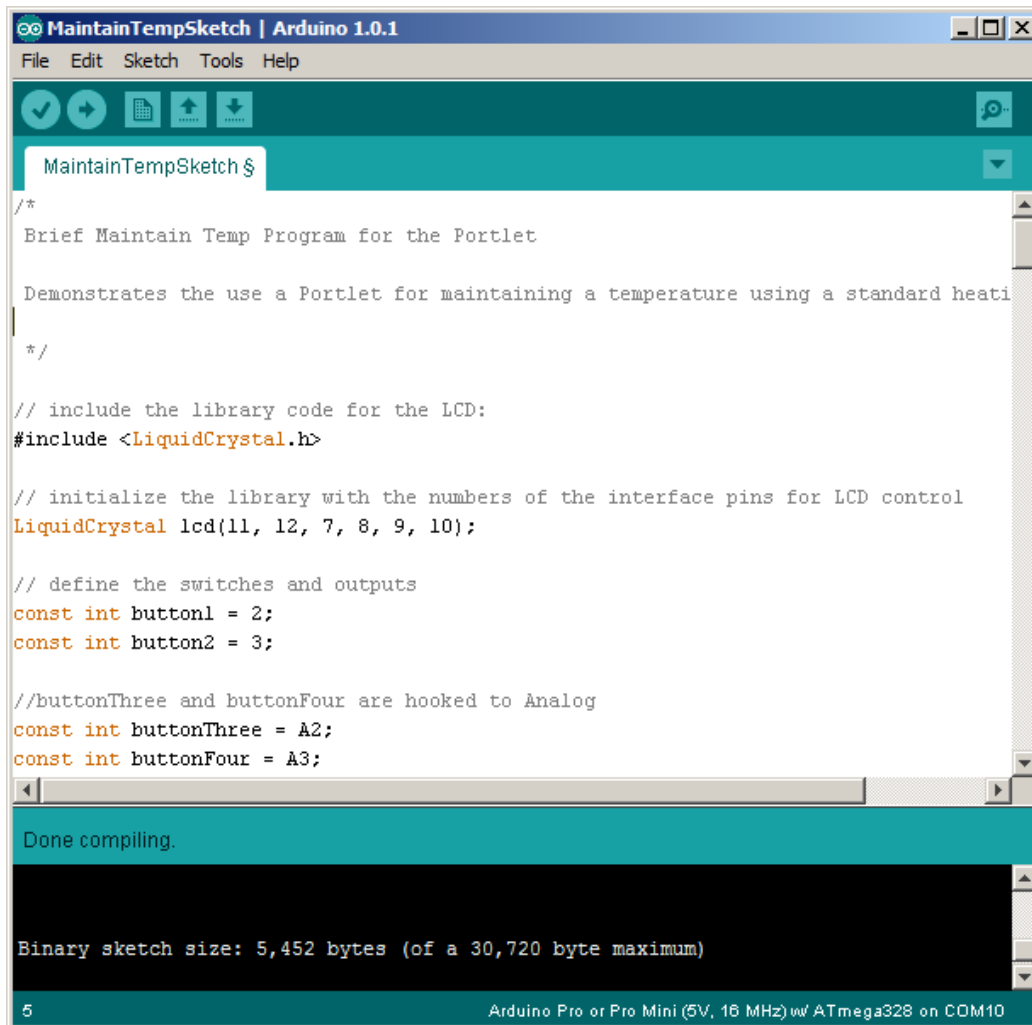
Programming the Portlet

Before you start, it's best to get the Portlet all ready and programmed. The Portlet is based off the popular Arduino platform and uses the Arduino IDE for programming. Here we give a brief example of how to program the Arduino, so if you want to learn more you should head over to the excellent Arduino home page at <http://www.arduino.cc/>

First off, plug your programming cable into the programming interface of the Portlet. The interface is the six header pins coming off of the Arduino Pro Mini build into the device. There is an access port in the covers on the right side of the Portlet. If you are using the Sparkfun 5V programming cable, the black wire goes on top for the correct orientation.



Open up the Arduino IDE and copy and paste the attached into the IDE. The pictures below show the Program using a different sketch, the Maintain Temp Sketch, but the process is the same.



The screenshot shows the Arduino IDE interface for a sketch named "MaintainTempSketch". The code in the editor includes a multi-line comment describing the program's purpose, followed by C++ code that includes the LiquidCrystal library, initializes an LCD object, and defines pin constants for buttons. The IDE's status bar at the bottom indicates that the sketch has been compiled successfully, with a size of 5,452 bytes.

```
MaintainTempSketch | Arduino 1.0.1
File Edit Sketch Tools Help

MaintainTempSketch $
/*
  Brief Maintain Temp Program for the Portlet

  Demonstrates the use a Portlet for maintaining a temperature using a standard heati
*/

// include the library code for the LCD:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins for LCD control
LiquidCrystal lcd(11, 12, 7, 8, 9, 10);

// define the switches and outputs
const int button1 = 2;
const int button2 = 3;

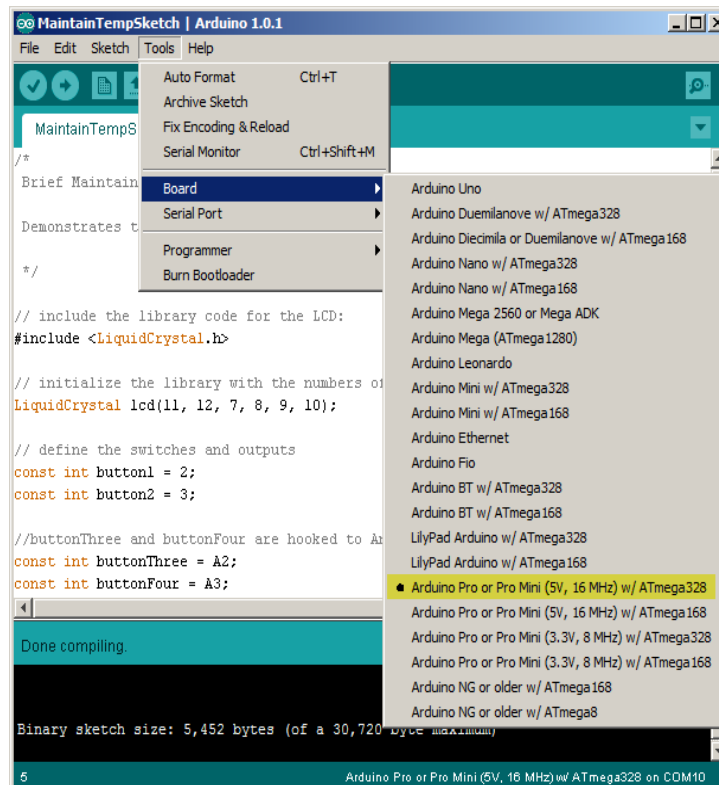
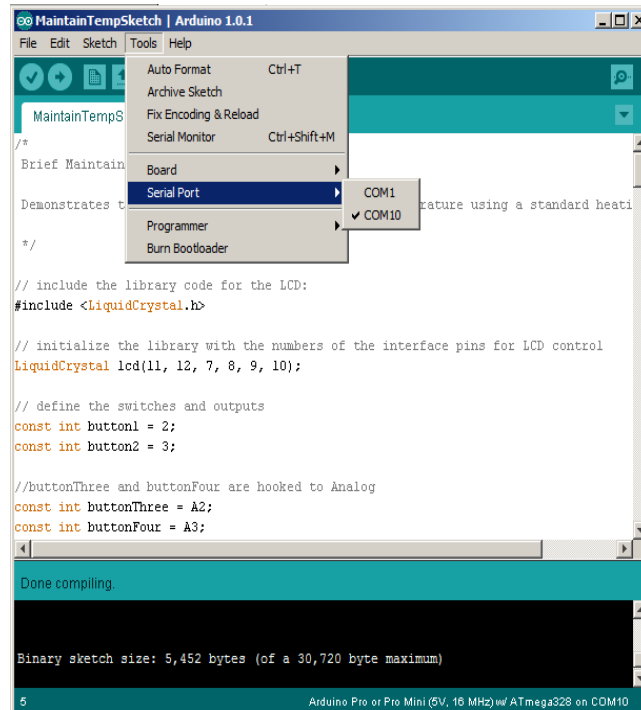
//buttonThree and buttonFour are hooked to Analog
const int buttonThree = A2;
const int buttonFour = A3;

Done compiling.

Binary sketch size: 5,452 bytes (of a 30,720 byte maximum)

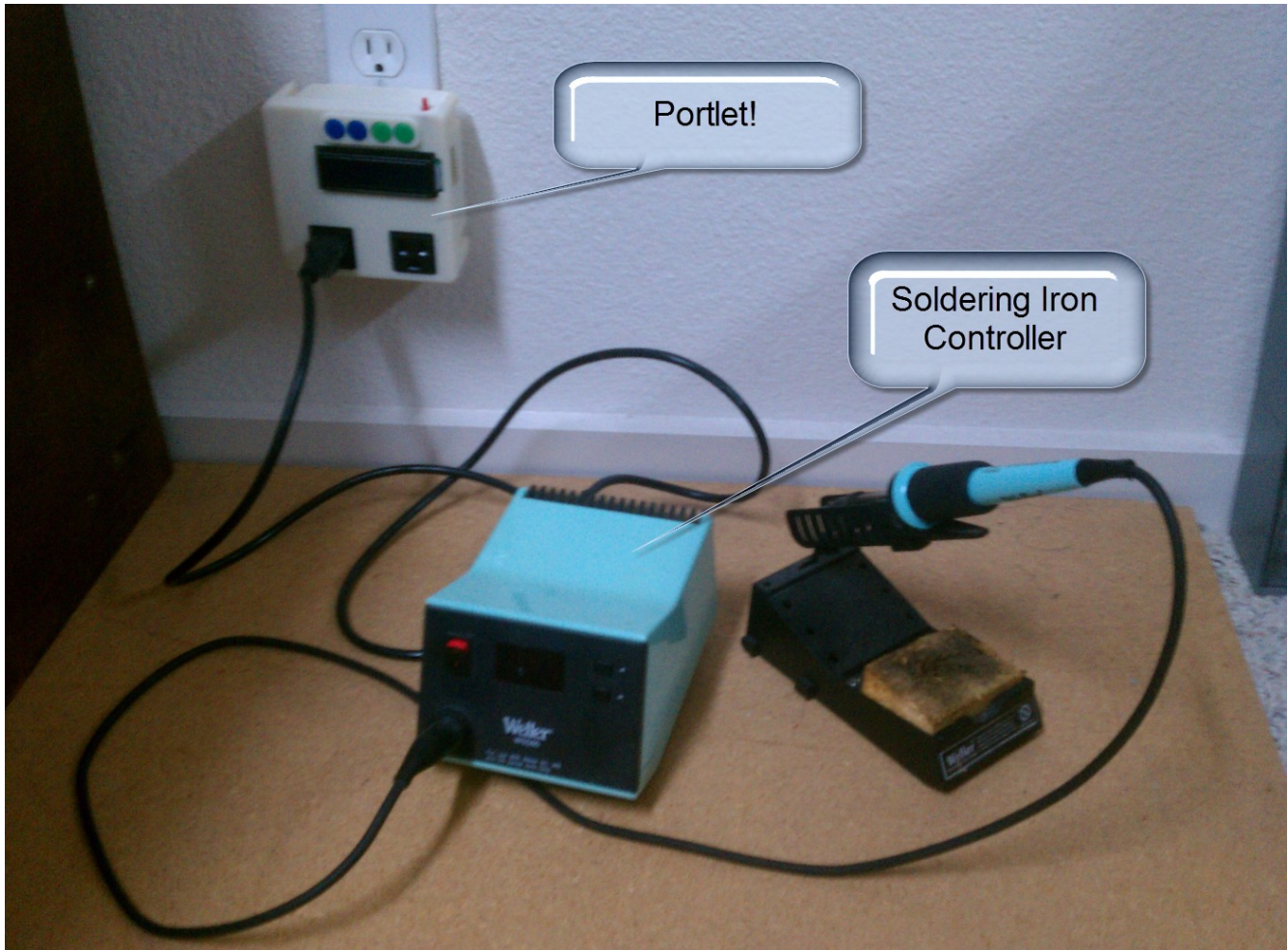
5 Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328 on COM10
```

Make sure you have the correct serial port selected for your interface and for the board choose the “Arduino Pro or Pro Mini (5V, 16MHz) w/ATmega328”



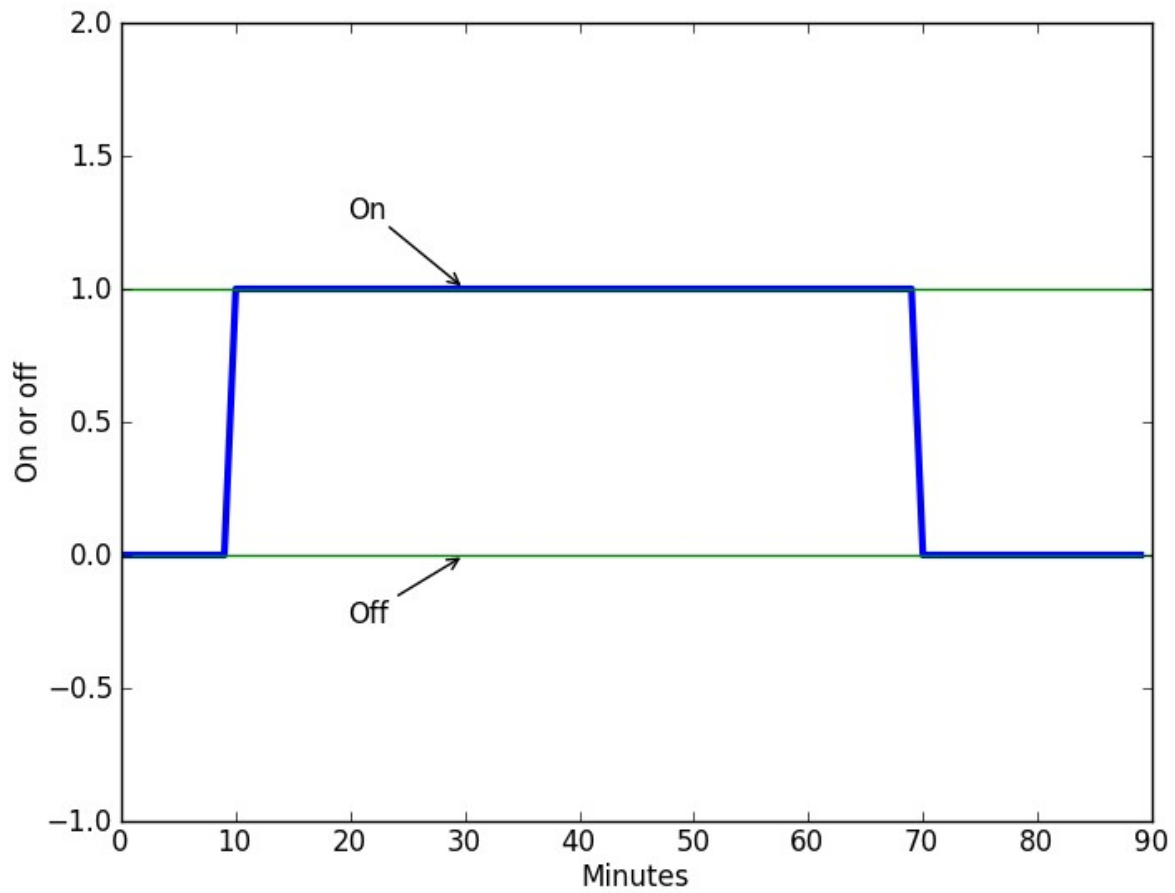
The Setup

Here you see the example soldering iron plugged into the Portlet which is in turn plugged into the wall outlet. Nice and simple.



Defining the outlet operation

If we are to graph this process, the basic operation profile of the outlet would look something like the following. On the Y axis a 1 indicates the outlet is turned on and a 0 indicates it is turned off. On the X axis, the timing can be totally up to you, the programmer.



Appendix A: Sample Arduino Code

```
/*  
Brief Outlet Shutoff Sketch for the Portlet  
  
Demonstrates the use a Portlet as a safety shutoff device  
  
*/  
  
// include the library code for the LCD:  
#include <LiquidCrystal.h>  
  
// initialize the library with the numbers of the interface pins for LCD control  
LiquidCrystal lcd(11, 12, 7, 8, 9, 10);  
  
// define the switches and outputs  
const int button1 = 2;  
const int button2 = 3;  
  
//buttonThree and buttonFour are hooked to Analog  
const int buttonThree = A2;  
const int buttonFour = A3;  
  
const int thermo1 = A0;  
const int thermo2 = A1;  
  
const int outlet1 = 6;  
const int outlet2 = 5;  
  
//the LCD backlight is used as feedback to the user  
const int backlight = 13;  
  
void setup() {
```

```
// set up the LCD's number of columns and rows:
lcd.begin(16, 2);

//define buttons as inputs and outlets as outputs
pinMode(button1, INPUT);
pinMode(button2, INPUT);

pinMode(outlet1, OUTPUT);
pinMode(outlet2, OUTPUT);
pinMode(backlight, OUTPUT);

//enable serial for troubleshooting
Serial.begin(9600);

}

void loop() {
  // prepare LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Press 1 ");
  lcd.setCursor(0, 1);
  lcd.print("to Start ");

  //make sure both outlets are off
  digitalWrite(outlet1, LOW);
  digitalWrite(outlet2, LOW);

  //turn the backlight on for ease of reading
  digitalWrite(backlight,HIGH);

  //do nothing loop until a button is pressed
  while(digitalRead(button1) == LOW);
```



```
//turn outlet 1 on
digitalWrite(outlet1, HIGH);
//start the timer
int time_in_minutes = 6;
for(int x = 0; x < time_in_minutes; x++){
  //print number of minutes elapsed
  lcd.clear();
  lcd.home();
  lcd.print("Num min: ");
  lcd.print(x);
  lcd.print(" ");
  //pause for 1 minute
  delay(6000);
  //for fun, send the number of minutes elapsed over the serial
  Serial.println(x);
}
}
```