

# Sensors and Actuators

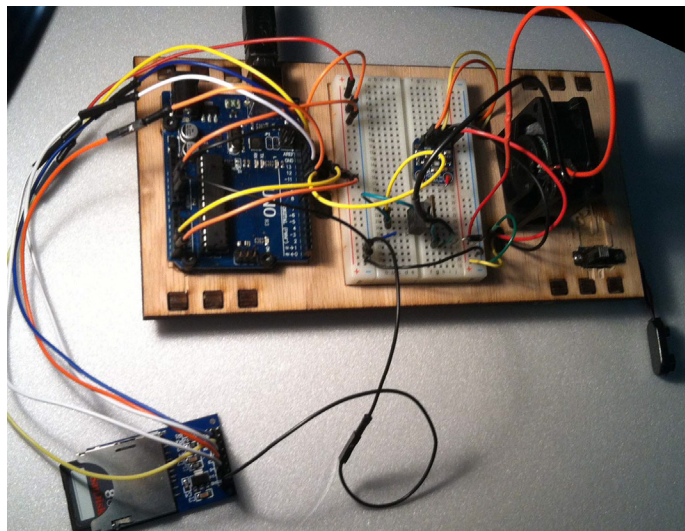
March 07, 2015

Stephen Wilkerson

The Army Research Laboratory,  
Aberdeen Proving Grounds

and

Frank Wattenberg,  
United States Military Academy,  
West Point



## Abstract

In this paper we begin with a simple project that will empower students to help solve our energy problems. They will build a “smart” system for controlling temperature. This project can be used as a stand-alone project, as part of a competition to build the best temperature controlled environment, or as a model for other projects. In addition to the basic Arduino starter kit you will need a small lamp and computer fan (details are given further on). To further simplify the programming and wiring we break this project into 3 separate components giving the details how to wire and program each part.

## 1 Background

In the past 10 years there has been an explosion of sensors, actuators and other output and storage devices available for small microprocessors like the Arduino and Raspberry Pi Micro-processors. In this paper we examine some mathematical applications that can be used with a sensor and a motor. In particular, we will make use of the Arduino Micro-processor coupled with a temperature

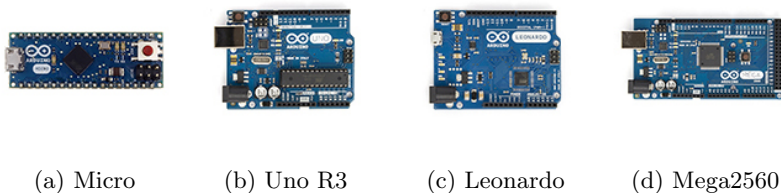


Figure 1: Some Arduino microprocessor Examples

sensor that is slowly heated up with a small lamp in an experiment. In addition to using a temperature sensor we will use a small fan to control the temperature of our sensor to within a specific range. This experiment has many applications like cooling of an enclosed area or regulating the temperature of a building, car, library, or a chemical process within specific ranges. Everything from car engines to coffee pots and toasters require specific operational ranges. In the past these devices were controlled using mechanical means. In the future, with the use of microprocessors, sensors, motors, and actuators, these functions might be controlled to far tighter specifications, increasing efficiency and conserving energy in many commonly used devices. Experiments like the one in this activity will hopefully act as a catalyst, to unlock the imagination of new students, instructors and entrepreneurs. Many ideas previously thought difficult are now achievable on a relatively small budget. Examples making use of small inexpensive microprocessors can be found in small pilotless drones, 3D printers, cell phones, and global positioning systems. We begin our activity with an overview of what will be needed to conduct a simple series of temperature control experiments.

There are two small sized micro processors that are dominating the world of small scale innovation: the Raspberry Pi and a micro-controller known as Arduino [1] [2]. Using these new microprocessors there seems to be no limits to the projects that can be devised. They range from simple gadgets for home and hobby use to whole new technologies that include: flying drones, unmanned robotic systems, 3D-printers, workshop tools, alarm systems, games and many other helpful instruments. Many of these projects are open source with a plethora of videos and other helpful tips readily available on the internet at no cost. This new era of do it yourself researchers and entrepreneurs has a strong educational background. For the experiment shown here we have chosen the Arduino Uno that can be purchased for less than 15 dollars. Not surprisingly, there are a number of compatible Arduino processors like the Micro, The Uno, the Leonardo and the Mega; see Figure 1 a,b,c, and d respectively for this suite of microprocessors devices.

The Arduino Micro is a micro-controller board based on the ATmega-32u4, developed in conjunction with Adafruit. It has 20 digital input/output pins (7 of which can be used as Pulse Width Modulation (PWM) outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an

In Circuit Serial Programming (ICSP) header, and a reset button. It has a form factor that enables it to be easily placed on a breadboard. The Arduino Uno is a micro-controller board based on the ATmega-328<sup>1</sup>. It has 14 digital input/output pins (6 of which can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button [3]. The Arduino Leonardo is a micro-controller board based on the ATmega-32u4. It has 20 digital input/output pins (7 of which can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the micro-controller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started . The Arduino Mega 2560 is a micro-controller board based on the ATmega-2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header and a reset button. Any of these boards can be used for the experiments we give here. However, as project complexity increases so does the need for computing power. A good example of this is the Mega and its derivative that are favorites of the Do It Yourself (DIY) 3D-printing movement.

Notwithstanding, there are also a number of shields that plug directly into the Arduino mother-board that gives a direct interface to the micro processor (no wiring required) for a variety of sensors. In addition, there are also a large number of compatible sensor and output devices from Ada fruit, Sparkfun, Arduino and others. We give a couple of examples here for reference in Appendix A Table A1 with potential suppliers. This is only a small example of the many sensor devices that can be found from a number of vendors. No less important are the many output devices that can also be interfaced with micro processors. Appendix A Table A2 provides a small subset of these for reference. Each of these devices usually come with an example or two complete with programming code that either extract data or make use of an output device. Once you get past the wiring, programming, and an example or two the only thing left is the potential applications and the mathematics that will be needed to make this happen. In this paper we will examine the steps needed to take a simple sensor gather data, do an analysis, and then solve a problem.

## 2 What's Required

Several things are needed for this experiment including the basic hardware, wiring and the software required to make the experiment work. In this first section we will examine specifically what is needed to make this all work, where to get these items, and the approximate cost. We will also look at how best to test the wiring connections and where to find help with problems. We will also examine the software needed to make the experiment run and its problems. Not all of these items are straight forward, but once mastered will enable

<sup>1</sup>High Performance, Low Power Atmel AVR 8-Bit micro-controller Family

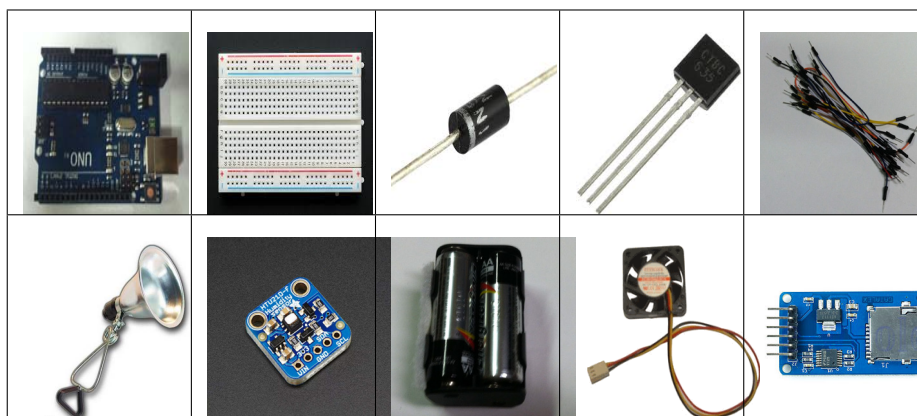


Table 1: Required materials for this sketch, from top left to right, Uno R3, breadboard, Diode, Transistor, push-wires, lamp, humidity and temperature sensor, battery pack, fan, sd card

students and instructors to explore the world from a new perspective of experimentation and discovery. Not surprisingly, the internet has a host of examples and can provide help when stuck. For example, there are numerous Arduino forums available where educators and experimentalists can exchange ideas and help one another over difficulties<sup>2</sup>. Information on specific projects, hardware, microcontrollers as well as education and teaching can all be found on line. In fact the Arduino and Raspberry Pi are some of the best supported products on the market. Whole new industries and market places have arisen from this new revolution in microprocessing.

## 2.1 Hardware

The following are the 10 basic items needed to conduct the experiment shown in this write up. Some basic knowledge of the Arduino micro processor, wiring of small non-complex circuits, searching the world wide web, and programming in "C" are assumed as a foundation for this project. However, you do not need to be an expert in these areas to conduct the experiment, but some basic knowledge will make this experiment go smoothly. The basic components required are listed below and shown above in Table 1. Item's with a "\*" next to them are part of the Standard Arduino Projects Book kit<sup>3</sup>.

<sup>2</sup><http://forum.arduino.cc/>.

<sup>3</sup>Items "\*" are in the standard kit: <http://arduino.cc/en/Main/ArduinoStarterKit>, or The Inland UNO Breadboard Kit for approximately \$15.00

- Arduino Uno R3<sup>4</sup>
- Bread Board<sup>4</sup>
- Diode<sup>4</sup>
- Transistor<sup>4</sup>
- Push pins<sup>4</sup>
- Battery<sup>4</sup>
- Printer cable<sup>4</sup>
- Fan \$3.00
- Lamp \$6.00
- Light Bulb \$5.00
- Temperature and Barametric Pressure Sensor \$15.00
- Micro SD card \$15.00(Optional)

These components are easy to find and many of the items required can be found at electronics stores like the Micro Center<sup>4</sup> or Fry's<sup>5</sup>. However, all items can also be found on line at Arduino<sup>6</sup>, Adafruit<sup>7</sup>, Spark Fun<sup>8</sup> and Amazon<sup>9</sup> to mention only a few locations.

## 2.2 Wiring

**Temperature and humidity sensor wiring:** We start with our wiring instructions by wiring the sensor and the Arduino Uno together. There already exists a wiring test that can be used to build a temperature recorder on adafruit's site<sup>10</sup>. We borrowed figure 9(from the adafruit site; footnote 6) as we will be using the exact same wiring for the Temperature and Humidity sensor for our experiment. The Adafruit example uses the UNO a4 and a5 pins to connect to the SCL and SDA pins on the HTU21D-F sensor respectively. A summary of the wiring is given in Appendix B.

**Fan wiring:** Motor wiring is important when using an Arduino microprocessor. For starters, the Arduino board does not have enough power to run a fan with a larger motor. Furthermore, due to the potential for back current it is important to include a diode to prevent current from flowing back into the UNO and damaging it. A basic project for motors can be found in the Arduino Projects book titled: "Motorized Pinwheel" project number 09. We will wire our project using the same pins that were used in the project Motorized Pinwheel in the Arduino UNO project workbook, but without the on off switch wired in. We give the diagram in figure 8 in Appendix C. Fortunately there is no conflict with

<sup>4</sup><http://www.microcenter.com/>

<sup>5</sup><http://www.frys.com/>

<sup>6</sup><http://www.arduino.cc/>

<sup>7</sup><http://www.adafruit.com/>

<sup>8</sup><https://www.sparkfun.com/>

<sup>9</sup><http://www.amazon.com/>

<sup>10</sup><https://learn.adafruit.com/adafruit-htu21d-f-temperature-humidity-sensor/wiring-and-test>

any of the pins required for the other components used in this experiment. The Arduino provides a variety of pin connections of various functions. These can be reviewed in the specifications for the Arduino Uno from the Arduino web site. A summary of the pins used to operate the fan (or a motor) are given in Appendix C.

**SD Card wiring:** The SD card is optional and not required to complete the experiments. We include it here for completeness. The card allows a student to run the experiment for a longer period of time and then to directly access the file with the data without the need of cutting and pasting from the Arduino serial window. The data can then be read into MatLab, Mathematica, or MS Excel for further analysis and plotting (See Mathematical section). We will show how this is done in the mathematical section. Figure 9 shows the basic wiring of the SD card to the Arduino Uno with one small exception. For our experiment we will use the 5V UNO bus to power the SD card reader. A summary of the connections are give in Appendix D.

### 2.3 Computer Code

In order to make use of the HTU21D-F Humidity and Temperature sensor, the SD card reader and the fan we will need some basic libraries. In all, we will need three libraries: (i.e. wire.h, adafruit HTU21DF.h and the SD.h libraries). To install these libraries on a specific system you will need to download them and then locate them on your computer. For simplicity I created a libraries file under the Arduino directory and placed the files there. Then to load those libraries into your Arduino you will need to start up a Arduino sketch and use the Arduino tab: Sketch>Import Library>Add Library and then add those to your system one at a time. No, you do not have to do this every time you write a new program. Do this once and then they are loaded into your system and can be accessed from any of your projects by simply including the correct code in the header file in your project program<sup>11 12 13</sup>. All of the code used for these initial Sketches can be found in Appendices. This sketch should work for any combination of sensors and actuators with the appropriate modifications for specific sensor and motor changes and is not limited to these specific choices.

## 3 Experiment

Our experiment requires some subtle adjustments of the fan and lamp. If the lamp is too close, the small fan that we are using will be insufficient to move enough air across the sensor to cool the sensor. On the other hand if the lamp is too distant the sensor will not heat up enough to make the experiment work. Hence, an initial trial is needed to make the experiment work. With the program

<sup>11</sup><http://arduino.cc/en/Reference/Wire>

<sup>12</sup><https://learn.adafruit.com/adafruit-htu21d-f-temperature-humidity-sensor/wiring-and-test>

<sup>13</sup><http://arduino.cc/en/Reference/SD>

loaded see what the ambient temperature is. Then turn on the lamp and make sure that the temperature increases. Once the ambient temperature in the room is determined, the fan can be set to come on when the temperature increases by a couple of a degrees Fahrenheit. Then turn on the fan ( $T > T_0 + 3$ ) to cool the circuit. Turn the fan back off when the temperature drops to a specific level ( $T < T_0 + 2.9$ ). Adjust the range of the light so that the temperature oscillates between a high temperature and a lower temperature. This is the basis of a temperature controlled process. There are many processes that can be controlled using similar mathematics. Here we just look at some of the basics for a control system. Using simple experimentation the students can force the temperature into a tightly controlled regime. However, this opens the door to some really good mathematical discussions on how best to control a process. Sampling the data at 1 second intervals it can be seen that the heating and cooling is very rhythmic in nature from Figure 5. This is the outcome desired so that further analysis is possible.

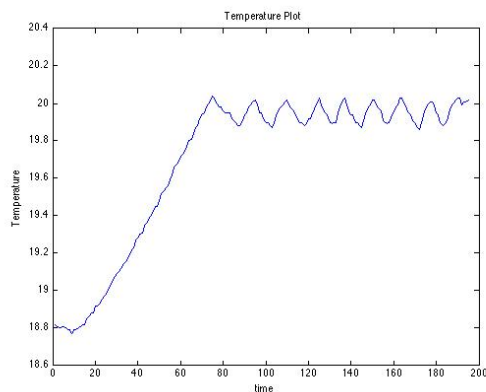
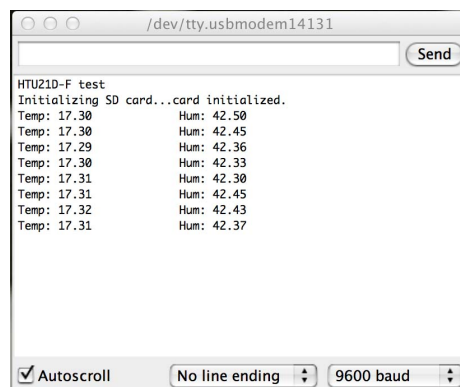


Figure 2: Experimental results at 1 second intervals

After the initial set up the adjustments required to get a good experiment in lamp position and fan location can be made while viewing the data on Arduino's serial display. This can be accomplished by selecting Tools>Serial Monitor tab from the Arduino controls while in the Arduino programming environment. The results will be displayed as in Figure 6. Watching this screen one can adjust the lamps distance to get the desired results. Once an acceptable arrangement is found, data can be recorded for further study.

## 4 Mathematics

This experiment provides a good example of how a process might heat up and cool off when being controlled by a heater and fan. Most processes that involve heat work on longer time scales. For example, it takes longer to heat up a car



```

/dev/tty.usbmodem14131
Send
HTU21D-F test
Initializing SD card...card initialized.
Temp: 17.30      Hum: 42.50
Temp: 17.30      Hum: 42.45
Temp: 17.29      Hum: 42.36
Temp: 17.30      Hum: 42.33
Temp: 17.31      Hum: 42.30
Temp: 17.31      Hum: 42.45
Temp: 17.32      Hum: 42.43
Temp: 17.31      Hum: 42.37
Autoscroll No line ending 9600 baud

```

Figure 3: Arduino serial display

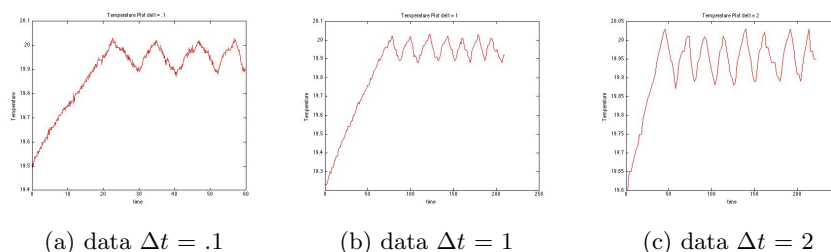


Figure 4: Time increments, from left to right .1, 1 and 2 sec. respectively

engine and potentially much longer to cool it off. The simple solution in this experiment, if we were trying to control the process to a vary narrow range, we would be to simply turn on the fan as soon as the temperature rises above a specific temperature and then turn it off again as soon as it drops below that same temperature. This is true due to the time frame and small scale of our experiment and the ability to rapidly heat and cool our sensor with the surrounding air. The way we got around this was to set a temperature range thus providing some latency in our process thus making it more realistic of a larger process that we might be trying to control. This additional constraint for our experiment allows us to cycle the temperatures on a larger scale. Another possibility is to change the sampling rate of the data. Different sampling rates are shown in figure 7a,b,c.

A brief discussion of how this data might be moved onto a scientific platform like MatLab or Mathematica is given here for completeness. In MatLab the import function was used. The original coding was modified to write a csv file by changing the write commands to the SD card. Of interest in controlling this process is the rate of change of the temperature (slope) and potentially the area under the curve:



$$slope = \frac{T_{i+1} - T_i}{\Delta t}$$

$$area = \sum_{i=1}^n T_i \Delta t$$

In the first case, the rate of change can be used as a measure of when exactly to turn on and off the cooling. In the latter, the area under the curve relates to the energy being used. In order to control these processes better two approaches can be taken. One would involve the controlling of the fan speed to force the system into tighter tolerances while the second might consider the timing of the fan's use. In the first case, the fan's speed could be tied to a proportional constant  $K_p$  dependent on the temperature feedback, while a similar constant  $K_d$  could be multiplied against the rate of change in the temperature, hence giving the system a proportional derivative PD control system. The process shown in this experiment is governed by Newton's law of cooling [4]:

$$\frac{dT}{dt} = -k(T - T_s)$$

The optimization of these constants  $K_p$ ,  $K_d$  is a subject within itself and is beyond the scope of this simple experiment. However, we include a simple MatLab simulink model here (Figure 5.) to stimulate future discussions.

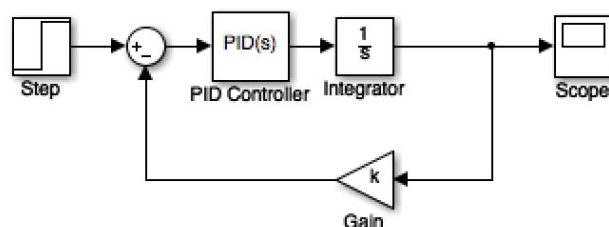


Figure 5: Simulink model of temperature control process

We start with the proportional constant  $K_p = .1$  and then use the tune function within Simulink to obtain  $K_p = .1$ ,  $K_I = .0017$ , and  $K_D = -.00044$ . The final results are shown in Figure 6.

## 5 Conclusion

In conclusion, it has been shown on a limited basis that data can be taken and used with an inexpensive micro-processor and a sensor to control a process. The

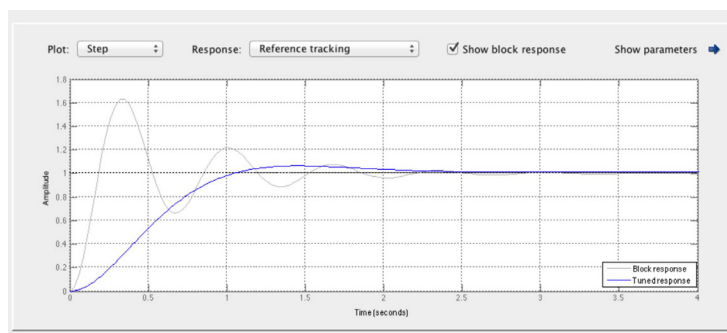


Figure 6: Simulink tuned model from the temperature control process

Arduino or Raspberry Pi micro processors coupled with a whole suite of sensors and actuators are now available for mathematical experimentation. For our limited example here we used the Arduino Uno and a temperature and humidity sensor to control the heat generated by a lamp at a specific location. Practical applications of these technologies, when coupled with the appropriate mathematical models, will spawn improvements in the fields of robotics, commercial applications, and home and hobby use. From camera stabilization to remote sensing and more all can be accomplished at minimal cost with the current supply of microprocessors. These new technologies will undoubtedly generate a host of new product areas and many other applications improving the human life style. Moreover, they offer a unique opportunity for mathematicians to apply their trade for scientific innovation and creativity.

## References

- [1] Charles Bell. *Beginning Sensor Networks with Arduino and Raspberry Pi*. APress, California 2013.
- [2] John Boxall. *Arduino Workshop. [A Hands-On Introduction with 65 Projects]*. No Starch Press, San Francisco, CA 2013.
- [3] Arduino Projects Book.  
*Arduino Projects Kit*
- [4] Wesley Day and Auburn Walker *Differential Cooling for Heating and Cooling*. 16 july 2002.

Picture	Supplier	Sensor Name	Description
	Ada Fruit	Global positioning System (GPS)	Adafruit Ultimate GPS Breakout 66 channel w/10 Hz updates
	Ada Fruit	Global positioning System (GPS) Shield	Adafruit Ultimate GPS Logger Shield - Includes GPS Module
	Ada Fruit	Inertial Measurement Unit (IMU)	Adafruit 10-DOF IMU Breakout
	Ada Fruit	Pixi Cam	A CMU Kickstarter project
	Spark Fun	Temperature Gauge	Digital Temperature Sensor Breakout - TMP102
	Spark Fun	Motion Sensor	PIR Motion Sensor

Table A1: Some Sensor Devices and Manufacturers.

## APPENDIX A






Picture	Supplier	Sensor Name	Description
	Spark Fun	Stepper Motor	Stepper Motor
	Mouser Electronics	Servo	Servo Motors Tinkerkit micro servo module
	Ada Fruit	Display	AdaFruit industries 772LCD shield kit
	Spark Fun	SD Card Reader	Sparkfun Breakout Board for MicroSD TransFlash
	Arduino	TWireless	APC220 Wireless RF Modules w/ Antennas / USB Converter for Arduino

Table A2: Some Output Devices and manufacturers.

## APPENDIX B

### Humidity and Temperature Sensor

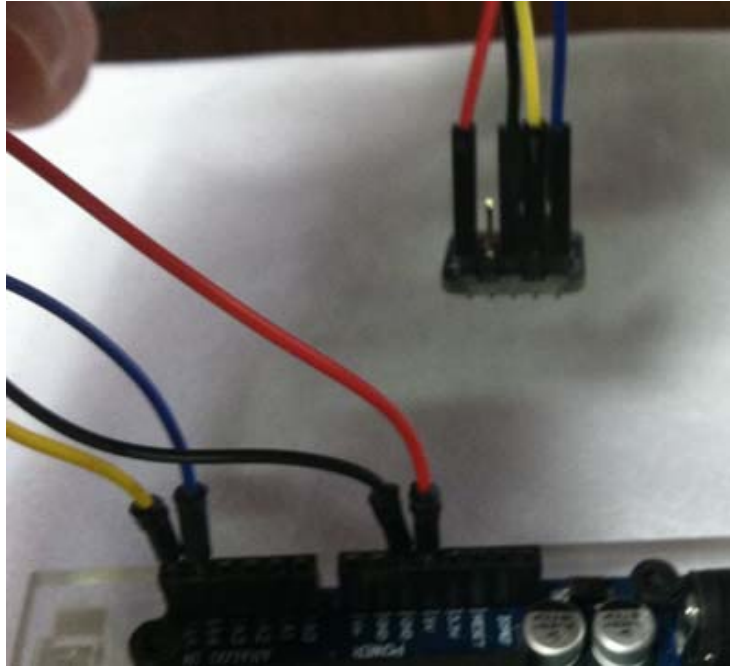


Figure 7: Wiring Diagram for HTU21-D Humidity and Temperature sensor

- HTU21D-F Sensor VIN to the UNO 5V.
- HTU21D-F Sensor GND to the UNO GND.
- HTU21D-F Sensor SCL to the UNO A5 pin.
- HTU21D-F Sensor SDA to the UNO A4 pin.

```
/*
 * This is an example for the HTU21D-F Humidity & Temp Sensor
 * Designed specifically to work with the HTU21D-F sensor from Adafruit
 * ----> https://www.adafruit.com/products/1899
 */
#include <Wire.h>
#include "Adafruit_HTU21DF.h"
// Variables
float temp1 = 0.0;
float hum1 = 0.0;
// Object
Adafruit_HTU21DF htu = Adafruit_HTU21DF();
// Setup
void setup() {
  Serial.begin(9600);
  if (!htu.begin()) {
    Serial.println("Couldn't find sensor!");
  }
}
```

```
    while (1);  
  }  
}  
/**Main program loop**/  
void loop() {  
  temp1 = htu.readTemperature();  
  hum1 = htu.readHumidity();  
  Serial.print("Temp: "); Serial.print(temp1);  
  Serial.print("\t\tHum: "); Serial.println(hum1);  
  delay(1000);  
}
```

## APPENDIX C

## Fan Wiring

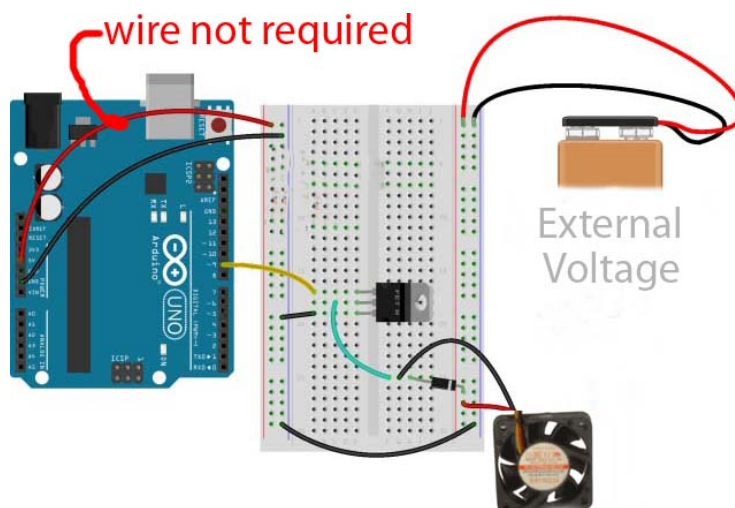


Figure 8: Diagram enabling the fan to be activated using Arduino pin 9

- External positive voltage source to bread board positive.
- External negative voltage source to bread board negative.
- Transistor pin 1 to the UNO pin 9 See Figure 2.
- Common ground both sides of bread board and Transistor middle pin.
- Fan wired between transistor pin 3 and Positive battery source. Diode between to prevent back flow (See Figure 2).

```
// No Libraries Required
// Set up constants
const int motorPin = 9; // the number of the motor pin
void setup() {
  // do nothing
}
void loop() {
  // Turn on the fan
  digitalWrite(motorPin, HIGH);
  // wait 5 seconds
  delay(5000);
  // Turn off the fan
  digitalWrite(motorPin, LOW);
}
```

## APPENDIX D

### SD Card Wiring

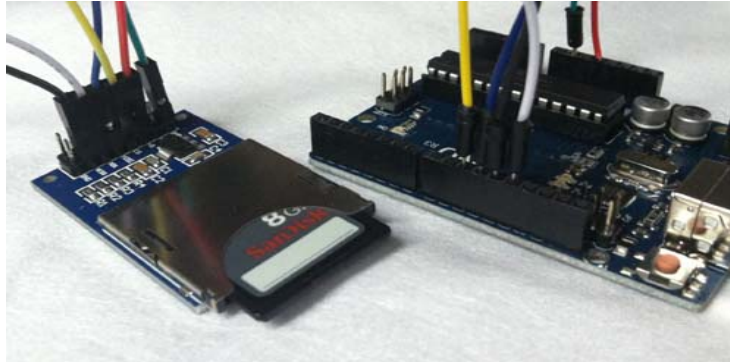


Figure 9: Diagram enabling The Sd card to be incorporated into the project.

- SD Card MOSI pin to Arduion Uno pin 11.
- SD Card MISO pin to Arduion Uno pin 12.
- SD Card CLK or SCK pin to Arduion Uno pin 13.
- SD Card CS pin to Arduion Uno pin 10. ...

```

//*****
// Include Libraries
#include <SD.h>
// Set up constants
const int chipSelect = 10;
int i=0;
// initial Setup
void setup() {
  Serial.begin(9600);
  // Initialize SD card
  Serial.print("Initializing SD card...");
  pinMode(10, OUTPUT);
  digitalWrite(10, HIGH);
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    return;
  }
  Serial.println("card initialized.");
}
void loop() {
  i=i+1;
  File dataFile = SD.open("Myfile.txt", FILE_WRITE);
  Serial.print("count"); Serial.print(i);
  dataFile.print("count"); dataFile.println(i);
  dataFile.close();
  // wait a second
  delay(1000);
}

```



## APPENDIX E

Complete all 3 Sketches combined

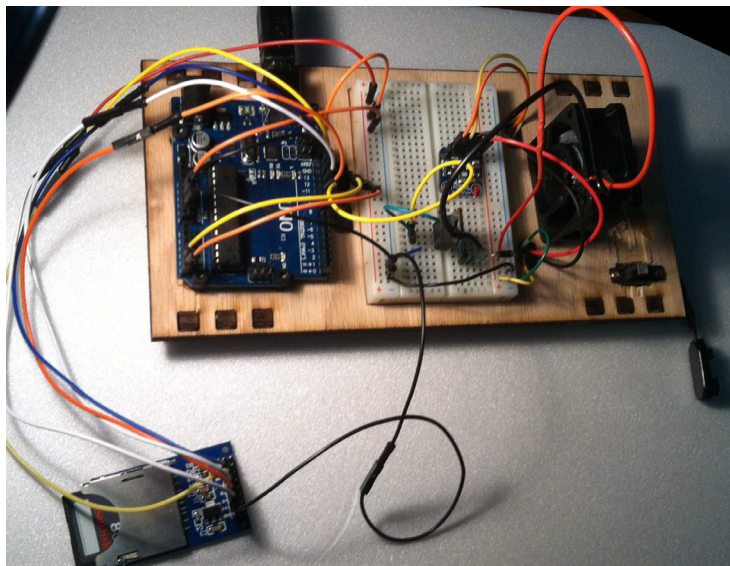


Figure 10: Complete project

```
/*
*****
This is an example for the HTU21D-F Humidity & Temp Sensor
with the SD card reader

For the HTU21D-F sensor alone from Adafruit please see:
https://www.adafruit.com/products/1899
*****
//
// Include Libraries
//
#include <Wire.h>
#include "Adafruit_HTU21DF.h"
#include <SD.h>
//
// Set up constants
//
float temp1 = 0.0;
float temp0 = 0.0;
float hum1 = 0.0;
float slope = 0.0;
float count = 0.0;
float t = 0.0;
int delT = 500;
const int motorPin = 9; // the number of the motor pin
const int chipSelect = 10;
int i = 0;
Adafruit_HTU21DF htu = Adafruit_HTU21DF();
void setup() {
  Serial.begin(9600);
  Serial.println("HTU21D-F test");
  pinMode(motorPin, OUTPUT);
  digitalWrite(motorPin, LOW);
  if (!htu.begin()) {
    Serial.println("Couldn't find sensor!");
    while (1);
  }
}
```

```
//
// Initialize SD card
//
Serial.print("Initializing SD card...");
pinMode(10, OUTPUT);
digitalWrite(10, HIGH);
if (!SD.begin(chipSelect)) {
  Serial.println("Card failed, or not present");
  return;
}
File dataFile = SD.open("Afile.txt", FILE_WRITE);
dataFile.print("count"); dataFile.print(";");
dataFile.print("time"); dataFile.print(";");
dataFile.print("temp"); dataFile.print(";");
dataFile.print("slope"); dataFile.print(";");
dataFile.println("humidity");
dataFile.close();
}
void loop() {
  temp1 = htu.readTemperature();
  i=i+1;
  t = t + (float)delt / 1000;
  if(i > 1){
    slope = (temp1 - temp0)/delt;
  }
  //
  // Gather Data
  //
  temp0 = temp1;
  hum1 = htu.readHumidity();
  //
  // Display info on Serial monitor
  //
  Serial.print(i); Serial.print(";");
  Serial.print(t); Serial.print(";");
  Serial.print(temp1); Serial.print(";");
  Serial.print(slope,5); Serial.print(";");
  Serial.println(hum1);
  //
  // Write data to file
  //
  File dataFile = SD.open("Afile.txt", FILE_WRITE);
  dataFile.print(i); dataFile.print(";");
  dataFile.print(t); dataFile.print(";");
  dataFile.print(temp1); dataFile.print(";");
  dataFile.print(slope,5); dataFile.print(";");
  dataFile.println(hum1);
  dataFile.close();
  // dataFile.print("Temp: "); dataFile.println(temp1);
  //
  // Turn on the fan when you're hot
  //
  if (temp1 > 20.0) {
    digitalWrite(motorPin, HIGH);
  }
  //
  // Turn Off Fan when Cool enough
  //
  else if (temp1 < 20.0) {
    digitalWrite(motorPin, LOW);
  }
  else
  {
    // do nothing
  }
  //
  // Continue collecting data at 1 second intervals till disconnected
  //
  delay(delt);
}
```