

Internet of Things Workshop

Lab 1

Getting Started with Arduino

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Reference |
| 10/5/2015 | Chmitch | 1.0 | Initial draft |
| 3/2/2016 | Stevebus | 2.0 | Updated to add command/control (turn on/off LED) |

Table of Contents

[Change Record 2](#_Toc431814134)

[Introduction 4](#_Toc431814135)

[Environment setup 4](#_Toc431814136)

[1.) Arduino Basics 6](#_Toc431814137)

[2.) Assemble your device and sensor 8](#_Toc431814138)

[3.) Build a sketch to read settings form the sensor 9](#_Toc431814139)

Introduction

This lab is focused on familiarizing you with the Arduino Uno device, and building an Internet of Things sensor using the device. This lab is the first in a series that walks through building an end-to-end Internet of Things prototype for doing temperature monitoring.

In this series of labs you will:

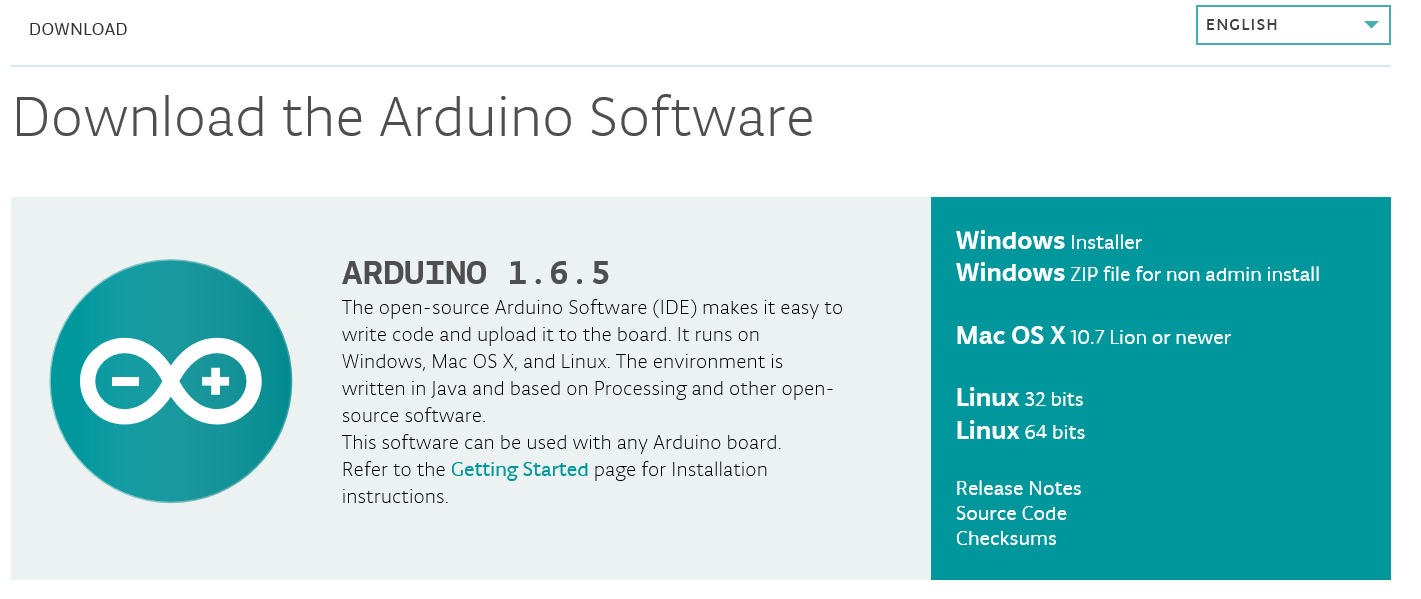
1. Assemble an Arduino Uno device for temperature monitoring using a prototype kit, and code and deploy a sketch using the Arduino IDE.
2. Write a gateway application (Universal Windows App) on a Raspberry PI to receive the serial data from the Arduino and send data to an Azure IoT Hub.
3. Configure Azure Stream Analytics jobs for gathering and aggregating streaming data for reporting purposes.
4. Build a Power BI dashboard for visualizing real-time and historical event data from the sensor.
5. Integrate the gateway app with the Azure IoT Suite Remote Monitoring pre-configured solution.

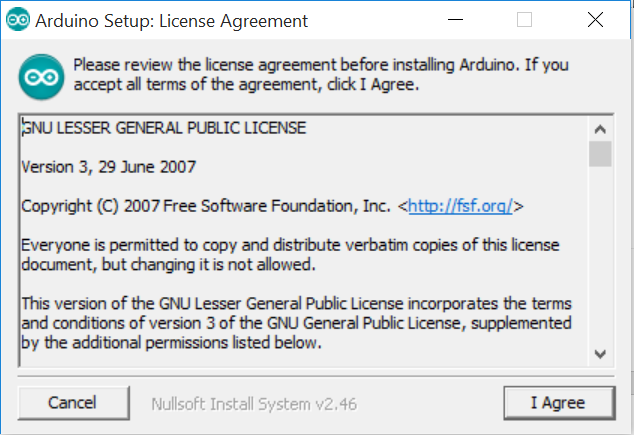
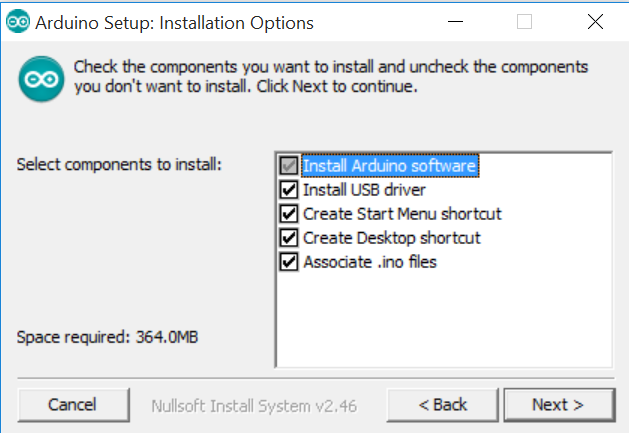
At the end of this lab you will have a functioning sensor capable of capturing ambient room temperature and humidity.

Environment setup

This lab module uses the Arduino IDE for development. **If this is not already installed on your workstation** please follow these steps for installation:

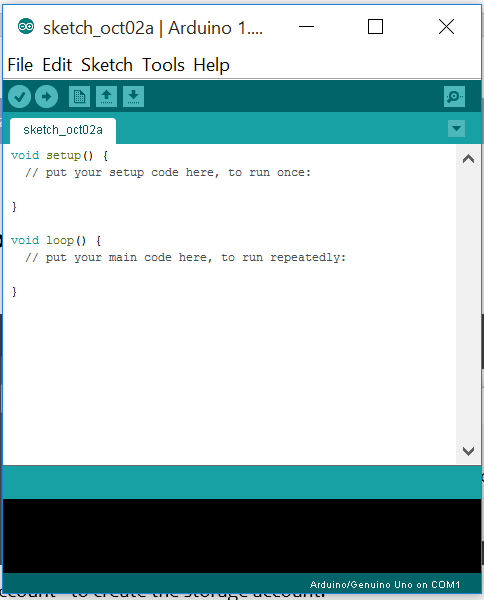
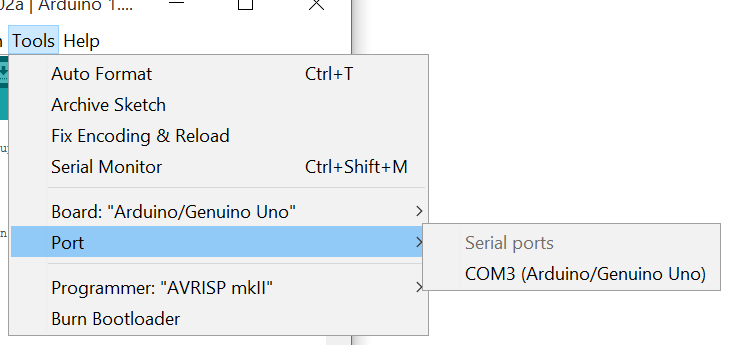
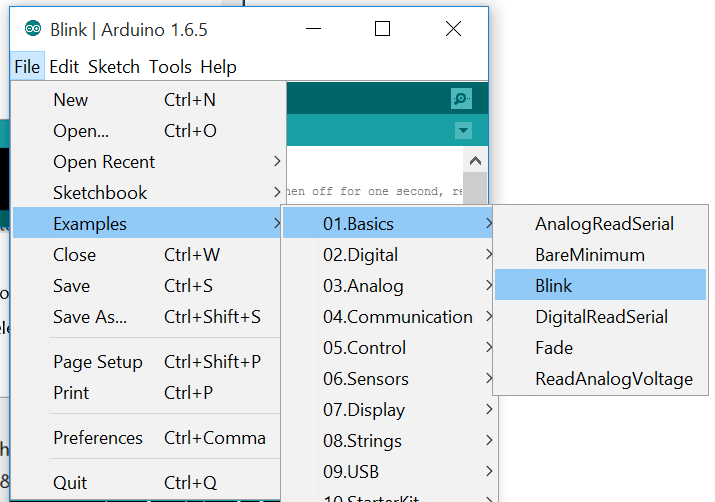
1. Using a web browser navigate to [www.arduino.cc](http://www.arduino.cc)
2. Click on the “Downloads” tab on the home page.
3. Click on “Windows” for the current windows installer.



4. Using explorer launch the installer and follow the default prompts for installation.  
   
(Note: Install USB driver is important as we’ll use this driver to communicate with the device and deploy code to it.)

1.) Arduino Basics

In this section you will familiarize yourself with key components of the Arduino IDE and validate your Arduino device is functioning properly:

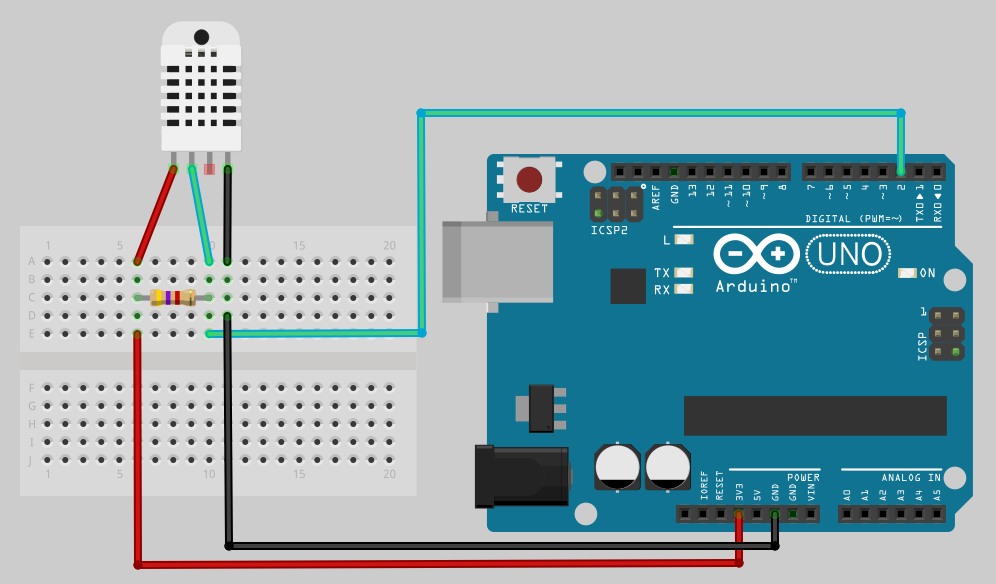
1. Launch the Arduino Desktop App. Upon launching you will be presented an empty project called a “sketch”.  
   
2. Connect the Arduino device to the workstation with the USB cable. (Note: the Arduino device can get power via either USB or an external power supply. For the purposes of this workshop we’ll be getting power via USB)
3. In the Arduino IDE you must select your device as your deployment target. Do this from the Tools -> Port menu:  
   
4. Now that the device is setup in the IDE, you can open and deploy a sample sketch. From the File -> Examples -> Basic menu open the “Blink” sketch.  
   
5. Click the deploy button  to load the sketch to the device. After the sketch has deployed look at your Arduino to validate you have a blinking light.

2.) Assemble your device and sensor

In this section you will familiarize yourself with key components of the Arduino IDE and validate your Arduino device is functioning properly:

Launch the Arduino Desktop App. Upon launching you will be presented an empty project called a “sketch”.

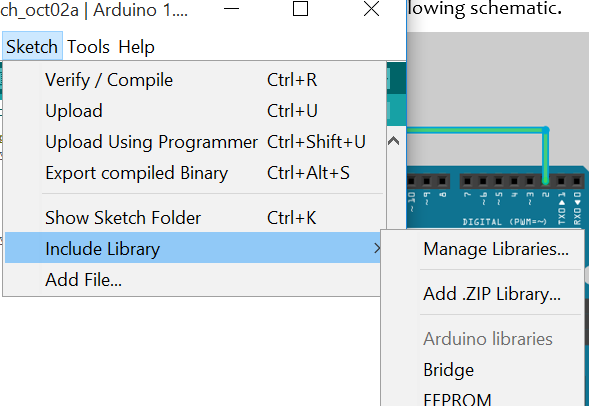
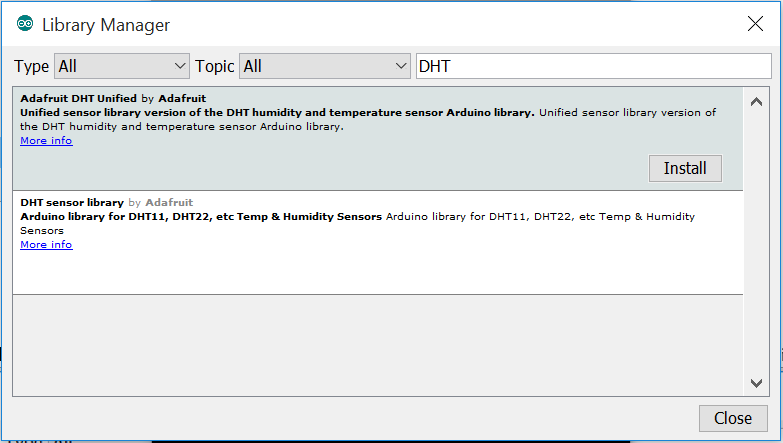
1. Disconnect the Arduino from your workstation. Note this step is very important to ensure there is no electric charge running through the device while we’re assembling.
2. With the provided jumper wires and breadboard assemble the Arduino using the following schematic. \*\* please note the diagram is logical and not to scale. The first and second pins cannot really be separated like shown \*\*

  
This diagram may seem complicated, so let’s deconstruct it a bit.

* 1. The black wire is the ground wire; it runs to the right most pin on the DHT sensor.
  2. The red wire provides power; it runs to the left most pin on the DHT sensor.
  3. The green wire is the signal wire it runs to the pin adjacent to the power lead.
  4. The power sensor doesn’t natively provide enough power to read directly, so we must draw some power and step it down with a resistor. (Don’t worry about the direction of the resistor they work either way)

3.) Build a sketch to read settings form the sensor

Now that the device has been assembled we can write some code to read the settings from the assembled sensor.

1. Plug your device back in to your workstation via USB.
2. In order to use the sensor we first need to download a library for simplifying communication with the device. In the Arduino IDE select “Manage Libraries” from the Sketch -> Include Library menu.  
   
3. From the library manager window search for “DHT”, select the ***second*** option “DHT sensor library by Adafruit” library, and click “Install”.  
   
4. When the install is complete close the Library Manager window.
5. Now it’s time to write some code. First we must include a reference and some initialization code for the DHT sensor. This includes referencing the installed module, defining which data pin we communicate on, and defining the sensor type (DHT22).  
   #include <DHT.h>  
   #define DHTTYPE DHT22  
     
   //Set’s the pin we’re reading data from and initializes the sensor.  
   int DHTPIN = 2;  
   DHT dht(DHTPIN,DHTTYPE);

String inputString = ""; // a string to hold incoming data  
boolean stringComplete = false; // whether the string is complete  
#define pinLED 13 // pin 13 is the onboard LED

1. Next we need to open the connection to the sensor, open the port for communication of sensor readings (we’ll be using a serial connection over USB), and finally begin DHT sensor readings. In the provided setup() procedure include the following:  
     
   //Tell the arduino we’ll be reading data on the defined DHT pin  
   pinMode(DHTPIN, INPUT);  
     
   //Open the serial port for communication  
   Serial.begin(9600);  
     
   //start the connection for reading.  
   dht.begin();

// we will be 'writing' to the pin ( vs. reading)  
pinMode(pinLED, OUTPUT);   
// start with the LED off  
digitalWrite(pinLED, LOW);

1. Finally, we need to capture the readings from the sensor and output them to the serial port. The DHT22 sensor is only rated to read data once every 2 seconds so we’ll need to include some code to prevent reading too frequently. We also add code to listen on the serial port for ‘commands’ from the gateway and turns the onboard LED on or off depending on the command. All the main application logic runs in the loop procedure which gets called after setup and runs as a never ending loop.  
     
   //declare variables for storing temperature and humidity and capture  
   float h = dht.readHumidity();  
   float t = dht.readTemperature(true);  
     
   //output data as humidity,temperature  
   Serial.print(h);  
   Serial.print(“,”);  
   Serial.println(t); //println includes linefeed

serialEvent(); //call the function to read any command in the serial buffer

// print the string when a newline arrives  
if (stringComplete) {

// turn LED on or off depending on command  
if(inputString == "OFF")  
 digitalWrite(pinLED, LOW);   
if(inputString == "ON")  
 digitalWrite(pinLED, HIGH);

// clear the string:   
inputString = "";  
stringComplete = false;

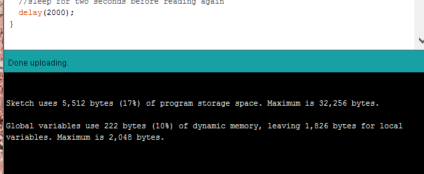
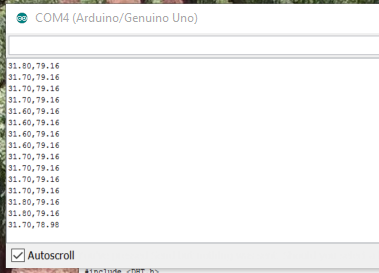
}

//sleep for two seconds before reading again  
delay(2000);

1. We need to add the serialEvent function to loop through the read data from the serial buffer until it hits a newline and returns the string read from the port.

void serialEvent() {

// while there is data to read in the buffer, read it  
 while (Serial.available()) {  
 // get the new byte:   
 char inChar = (char)Serial.read();  
 // add it to the inputString. if it's a newline, bail as that completes the message  
 if (inChar == '\n') {  
 stringComplete = true;   
 }  
 else  
 inputString += inChar;   
 }  
}

1. Now that the code is complete you can deploy it to the device using the deploy button . A successful deployment will display the following:  
   
2. To see that your sketch is running correctly use the serial monitor from the Tools menu. This will display the data that the Arduino is sending over the COM port.  
   
3. In the serial monitor, on the bottom right, make sure “newline only” is chosen. Then type “ON” into the input box on the top, hit “SEND” and ensure the onboard LED lights up. Type “OFF” and hit SEND and make sure the LED turns back off
4. Congratulations, you’ve built and programmed your sensor. For more information on the DHT22 sensor see the adafruit website: <https://learn.adafruit.com/dht/overview>