**ACAT Arduino Proximity Sensor User Guide (v1.00)**

**Table of Contents**

[**Introduction** 2](#_Toc20318999)

[**Setting Up the Arduino Proximity Sensor** 2](#_Toc20319000)

[**To get started the following materials are needed:** 3](#_Toc20319001)

[**Physically Putting the Arduino Sensor Together and Programming It** 3](#_Toc20319002)

[**Setting Up and Using the Windows Interface App (AcatActuator.exe)** 7](#_Toc20319003)

[**Setting Up the AcatActuator App** 7](#_Toc20319004)

[**Explanation of what the AcatActuator.exe Application Does** 10](#_Toc20319005)

[**Explanation of the Components in the Main Window** 10](#_Toc20319006)

[**Developer’s Guide** 12](#_Toc20319007)

[**Arduino Code Overview** 12](#_Toc20319008)

[**Windows Interface App (AcatActuator) Code Overview** 12](#_Toc20319009)

[**Licensing Information** 12](#_Toc20319010)

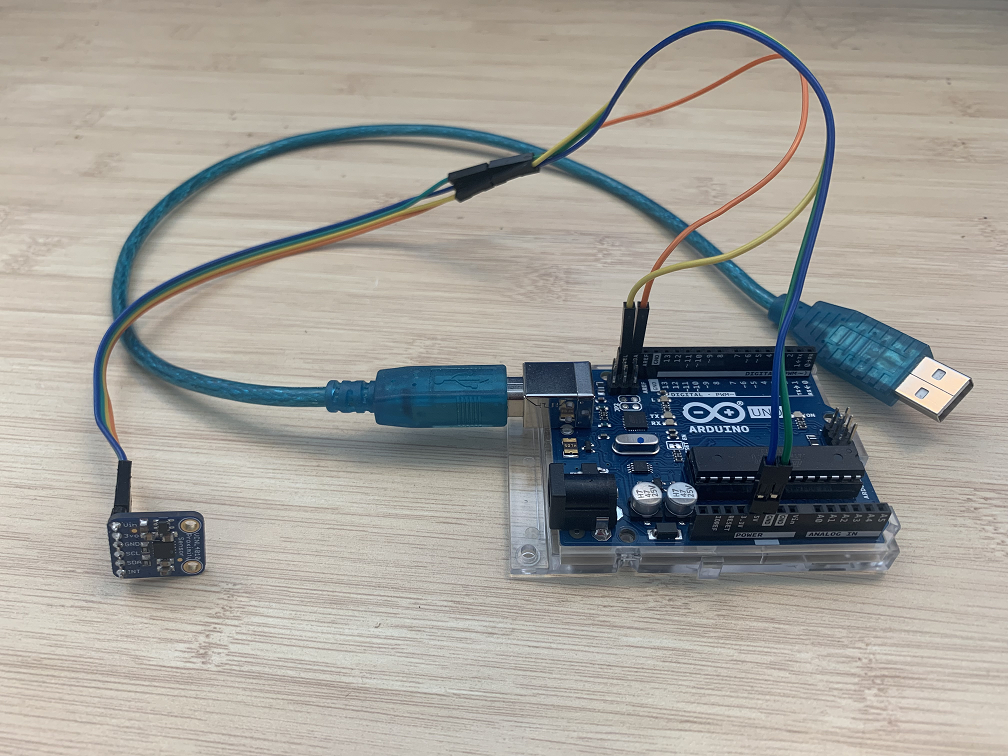
# **Introduction**

The Assistive Context Aware Toolkit (ACAT) supports a variety of sensors including ACAT Vision (camera based sensor) and most commercially available “off-the-shelf” sensors. However, for some users, using the camera based sensor may not be feasible due to their condition and the commercially available sensors may be too expensive. To bridge this gap, Intel Labs has developed an Arduino based proximity sensor solution that is also open source. If the potential ACAT user has financial constraints that prevent them from purchasing "off-the-shelf" sensors and has someone with minimal technical experience to help set up the Arduino proximity sensor, then this system may be the best solution for them. Additionally, the system is open source so developers can build on top of the system by adding new sensors, features, algorithms, etc.

The VCNL4010 proximity sensor used in the system best operates by detecting movements close to the sensor (within ~4cm). Some examples of ideal movements that could be detected by the sensor include: hand / finger movements, facial movements, and leg / foot movements. Set up of the sensor requires some soldering (about a couple minutes), finding someone that already knows how to do so is recommended. The sensor has holes for screws so it is flexible in terms of mounting options. Please see below video for a demo of the system.

<https://www.dropbox.com/s/ztshahmd5rkafrl/Arduinoproximitysensor_demo1.mp4?dl=0>

# **Setting Up the Arduino Proximity Sensor**

****

**Figure 1: Arduino Proximity Sensor System**

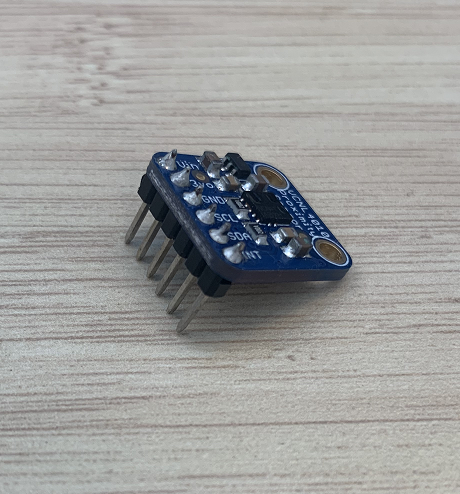
## **To get started the following materials are needed:**

1. Computer with Windows 7 or higher
2. Arduino Uno Board (~$22)
   1. <https://store.arduino.cc/usa/arduino-uno-rev3>
   2. The Arduino Uno is recommended and the only board that has been tested with this system. You should be able to use other Arduino boards but it needs to have an I2C interface to connect to the proximity sensor. Please check so before purchasing
3. VCNL4010 proximity sensor (~$8)
   1. <https://www.adafruit.com/product/466>
   2. We recommend the VCNL4010 proximity sensor from Adafruit. It is a breakout board meaning no additional components are needed. You only need to solder the header pins to the board
4. Wires to connect the proximity sensor to the Arduino (~$7)
   1. <https://www.amazon.com/Elegoo-EL-CP-004-Multicolored-Breadboard-arduino/dp/B01EV70C78/ref=sr_1_1_sspa?keywords=breadboard+cables&qid=1564884004&s=gateway&sr=8-1-spons&psc=1>
   2. We recommend “jumper wires” as they are easy to connect and can be linked together to form connections of various lengths
5. Soldering kit (if you don’t already have one)
   1. <https://www.amazon.com/ANBES-Soldering-Iron-Kit-Electronics/dp/B06XZ31W3M/ref=sr_1_3?keywords=soldering+kit&qid=1564884293&s=gateway&sr=8-3>
   2. This is the only thing that requires some technical know how to use. The soldering is minimal (takes only a couple minutes) but we recommend finding someone who already know how to do this as it can be dangerous since you are dealing with a high temperature tool. If you want to learn how to solder, please see the below YouTube tutorial
      1. <https://www.youtube.com/watch?v=Qps9woUGkvI>
6. Arduino case (~$10. Optional)
   1. Not necessary but nice to have to protect the Arduino

## **Physically Putting the Arduino Sensor Together and Programming It**

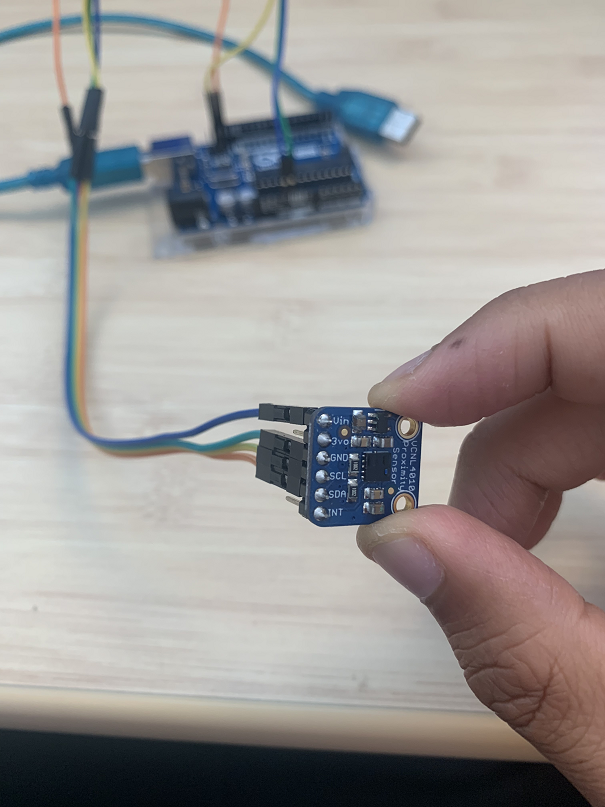
The steps below will show you how to put the Arduino sensor together and program it

1. Solder the header pins to the VCNL4010 sensor
   1. We recommend finding someone who already knows how to solder if you don’t know how. Please see the above note (#5 in previous section) about soldering if you haven’t read it already
   2. Make sure the long side of the header pins is extruding from the sensor

****

**Figure 2: VCNL4010 Proximity Sensor with Soldered Header Pins**

1. Connect the jumper wires to the proximity sensor pins
   1. You’ll need 4 wires which need to be connected to the pins labelled “Vin”, “GND”, “SCL”, “SDA”
   2. Please see below pic for the required connections

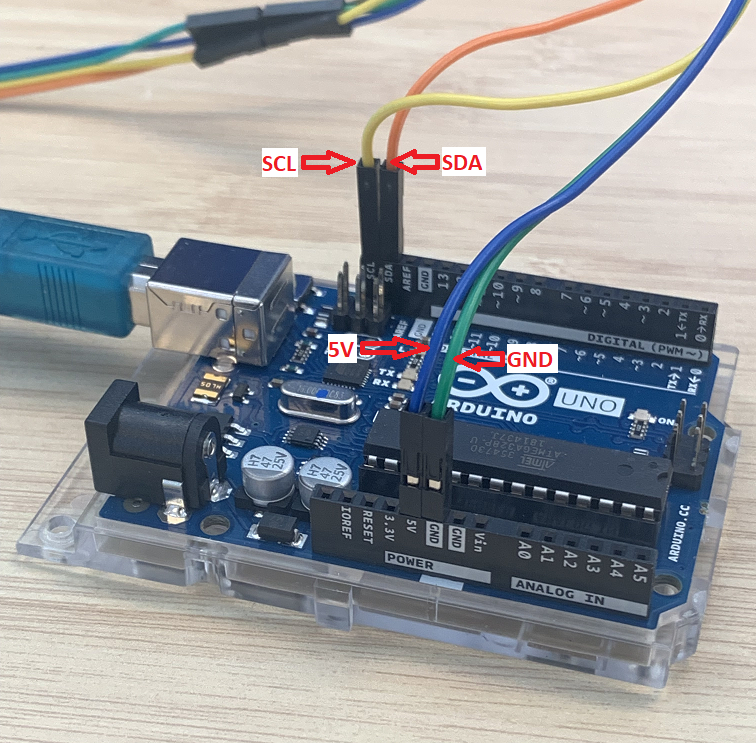
****

**Figure 3: Jumper Cables Attached to Necessary VCNL4010 Pins**

* 1. You can connect multiple jumper wires together to form a longer connection between the Arduino and the proximity sensor. In the pic of the entire system above (Figure 1) we connected 2 jumper cables for each connection (total of 8 wires)

1. Connect the proximity sensor to the Arduino
   1. Make sure the Arduino is not plugged in and receiving power before doing this step
   2. The following connections should be made from the proximity sensor to the Arduino. Please see the below table and picture

|  |  |
| --- | --- |
| **VCNL4010 Proximity Sensor** | **Arduino Uno** |
| Vin | 5V |
| GND | GND |
| SCL | SCL |
| SDA | SDA |



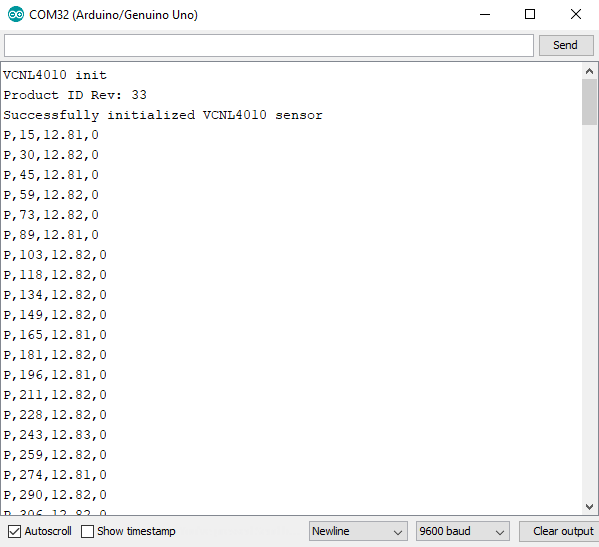
**Figure 4: Necessary I2C Connections From VCNL4010 to Arduino**

1. Install the Arduino Software IDE on the Windows PC and flash example program
   1. This will allow you to program the Arduino sensor
   2. Follow the steps on the Arduino website to install the Arduino Software IDE. Follow the suggested guidelines
      1. <https://www.arduino.cc/en/guide/windows>
   3. After installing the IDE, follow the “Getting Started” steps to install the Arduino drivers and flash an example program to familiarize yourself with that process. The example program will blink an LED on the board
      1. “Getting Started” steps for the Arduino Uno - <https://www.arduino.cc/en/Guide/ArduinoUno>
2. Download the source code for the Arduino sensor
   1. **(Temporary location)** [**https://www.dropbox.com/sh/dwpia2tqat4qxut/AADwI1wHcuhwGEa7rjbtu8NMa?dl=0**](https://www.dropbox.com/sh/dwpia2tqat4qxut/AADwI1wHcuhwGEa7rjbtu8NMa?dl=0)
   2. Copy the contents of the “libraries” folder downloaded above to the Arduino libraries folder in My Documents on your machine which should be “C:\Users\yourWIndowsUserName\Documents\Arduino\libraries”
      1. Where your “yourWIndowsUserName” is your windows user name
3. Program the Arduino sensor with the downloaded source code
   1. Plug in the Arduino sensor into the computer through usb if it is not plugged in already
   2. In the Arduino Software IDE, find and open the source code by doing “File -> Open” and searching for and opening the “ACAT\_Arduino\_Proximity\_Sensor.ino” file downloaded in Step #5
   3. You should see the Arduino program loaded like below

****

**Figure 5: ACAT\_Arduino\_Proximity\_Sensor Program Loaded in Arduino IDE**

* 1. Press the “Upload” button in the top left. The program is now being written to the Arduino
  2. After it is complete, in the toolbar do “Tools -> Serial Monitor”. If all goes well, you should see the proximity data being printed. Please confirm this is the case before moving on

****

**Figure 6: Serial Monitor in Arduino IDE Successfully Displaying Proximity Sensor Data**

# **Setting Up and Using the Windows Interface App (AcatActuator.exe)**

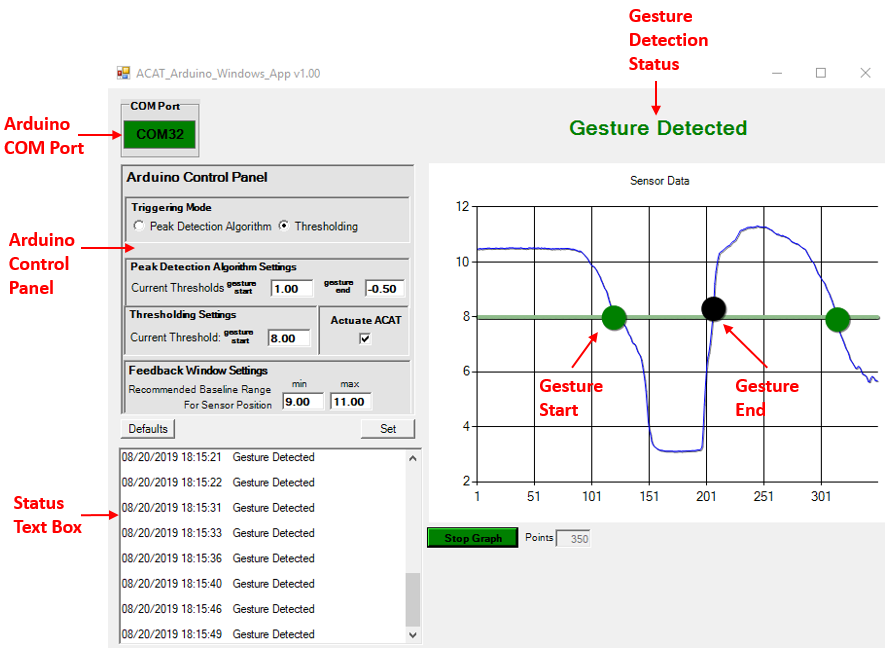
## **Setting Up the AcatActuator App**

The steps below will show you how to use the Windows interface app (AcatActuator.exe) that communicates with the Arduino and triggers ACAT when a gesture is detected

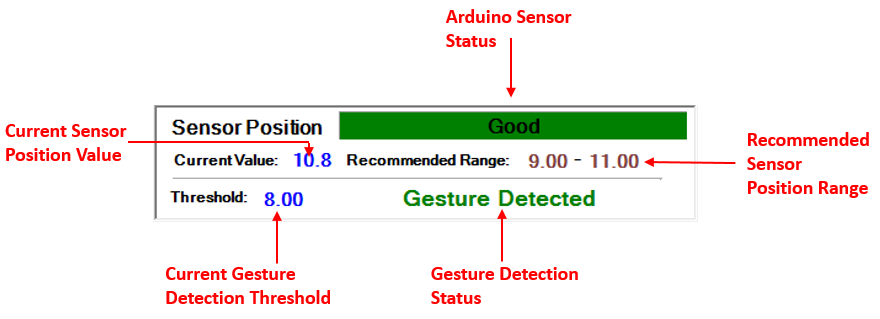
1. Download the AcatActuator program folder and launch the AcatActuator.exe
   1. **(Temporary location)**

[**https://www.dropbox.com/sh/qx6sfmfc0nkpgzo/AADPAv1z\_spcsGhNhoNUnj7fa?dl=0**](https://www.dropbox.com/sh/qx6sfmfc0nkpgzo/AADPAv1z_spcsGhNhoNUnj7fa?dl=0)

1. You should see two windows appear. One is the main control window where you can change the Arduino sensor settings and see the data coming in. The other window which should appear in the bottom right is the “feedback window” which always shows the Arduino sensor status and recommended position. The feedback window should appear on top of all other windows except ACAT ones

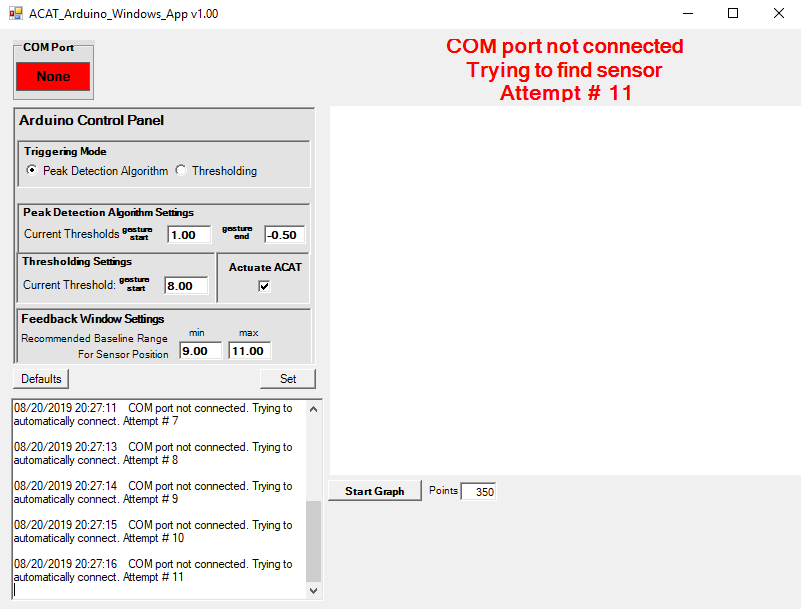
****

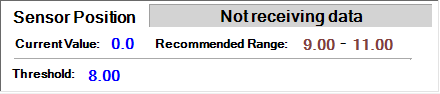
**Figure 7: Main AcatActuator.exe Control Window**

****

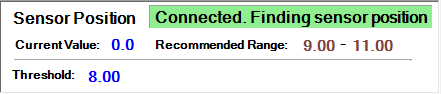
**Figure 8: AcatActuator.exe Feedback Window**

1. The AcatActuator app should automatically find and connect to the Arduino sensor. It attempts to do so every few seconds. Check the COM port and status message in the main window and feedback window to make sure that is has successfully done so before moving on.
   1. If it can’t find the Arduino, make sure that you have installed the Arduino drivers found in the “Getting Started” guide from the Arduino website (step #4c in the previous section)



****

**Figure 9: Main Window and Feedback Window Showing No Connection to Arduino**

****

**Figure 10: Feedback Window Showing Successful Connection to Arduino**

## **Explanation of what the AcatActuator.exe Application Does**

Once the connection between AcatActuator.exe and the Arduino is established, the settings displayed in the main window are sent to the Arduino. These settings are saved in “App\_Settings.xml” in the AcatActuator.exe program folder. Whenever the “Set” button is pressed, if changes to the settings have been made, then these settings are sent to the Arduino and saved in the “App\_Settings.xml” file. The “Defaults” button loads the default settings from “App\_Settings\_DEFAULT.xml” and sends them to the Arduino.

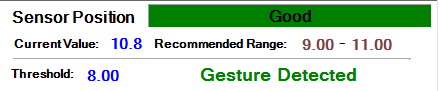
When a gesture is detected on the Arduino side, it sends messages to the AcatActuator app which then parses those messages and triggers ACAT accordingly. Make some movement in front of the sensor, it should display “Gesture Detected” message and if ACAT is opened, it should trigger it.

There is an option to display the data coming from the Arduino sensor. To enable / disable this feature, press the Start Graph / Stop Graph button. We only recommend showing the data when calibrating the sensor position and finding good thresholds, otherwise the graphing feature slows down the computer. It is disabled by default.

The feedback window on the bottom right of the screen gives the status of the connection to the Arduino, recommendations on sensor placement, and when gestures are detected. When there is no movement in front of the sensor, the app tries to detect the sensor’s distance from the user’s body and recommends to either move it closer or further away.

****

**Figure 11: Feedback Window Recommending User to Move Sensor Away From Body**



**Figure 12: Feedback Window Showing Good Sensor Position and Gesture Detected**

## **Explanation of the Components in the Main Window**

* **COM Port** – The COM port of the Arduino which the AcatActuator app is connected to. Displays “None” if it can’t find the Arduino COM port
* **Triggering Mode** – How the Arduino processes data and decides to send gesture start / end messages
  + **Peak Detection Algorithm** – This mode basically senses any type of movement in front of the sensor. It is the same algorithm implemented in the sensor Stephen Hawking used. Use this mode if the user can’t make controlled movements (close to the sensor then away from the sensor) very well.
  + **Thresholding (Default)** – This mode is a simple mode where if the value of the proximity sensor passes a certain threshold, then a gesture is detected. Use this mode if the user can make controlled movements very well.
* **Peak Detection Algorithm Settings** – Controls how sensitive the peak detection algorithm is to movement
  + **Gesture Start** – Threshold for when the output of the algorithm sends gesture start message. Increase this number if it is too sensitive to movement. Increase this number if it is not picking up the movements. The default value of 1.00 is already pretty sensitive and should pick up all movement in front of the sensor within about 3.5cm
  + **Gesture End** – Threshold for when the output of the algorithm sends gesture end message. Recommend not changing this number
* **Thresholding Settings** – Controls how far the sensor detects a gesture in this mode
  + **Current Threshold** – Controls the distance at which movement triggers a detected gesture. The lower the value, the closer you need to be to the sensor to trigger a gesture. The higher the value, the further the triggering threshold is. You can figure out a good threshold by enabling graphing of the data and seeing what the resting value is and what the value is when a trigger is attempted. However, don’t leave graphing enabled all the time as it slows down the performance on the computer
* **Feedback Window Settings** – Controls the recommended sensor position range as displayed in the feedback window under “Recommended Range”. Once a good sensor position is determined for the user, then these values can be changed if needed.
  + **Min** – The minimum value of the “Recommended Range”
  + **Max** – The maximum value of the “Recommended Range”
* **“Defaults” button** – Loads the default settings as saved in “App\_Settings\_DEFAULT.xml” in the AcatActuator folder
* **“Set” Button** – Sends all settings displayed in the main window to the Arduino and saves them to “App\_Settings.xml” in the AcatActuator folder
* **“Start Graph” Button** – Enables / disables graphing of the data coming from the Arduino. Useful to find a good sensor position and threshold but disable after you are done since leaving it on will slow down the performance of the computer. Graphing is disabled by default.
  + **“Points” text box** – Control how many data points from the Arduino are graphed

# **Developer’s Guide**

The entire code base of the ACAT Arduino proximity sensor is open source so developers can build on top of it (add new sensors, features, algorithms, etc.). Download the codebase which includes the Arduino code and Windows interface app (AcatActuator.exe) here

**(Temporary location)**

[**https://www.dropbox.com/sh/ts2ujae6y8g79pe/AABtvb-OYNp1PHUP1sgoYgD3a?dl=0**](https://www.dropbox.com/sh/ts2ujae6y8g79pe/AABtvb-OYNp1PHUP1sgoYgD3a?dl=0)

The Arduino code is in C++ and any text editor or IDE compatible with the C++ programming language can be used to modify that code.

The Windows Interface App (AcatActuator) was implemented in C# using Microsoft Visual Studio Professional 2017 with Microsoft .NET framework 4.7

## **Arduino Code Overview**

ACAT\_Arduino\_Proximity\_Sensor.ino – Main Arduino “sketch file”. Loaded into Arduino Software IDE and connects the proximity sensor logic to the AcatActuator logic

VCNL4010 – VCNL4010 proximity sensor driver

Actuator – Processes the proximity data and sends messages to and receives messages from the Windows interface app

cppQueue – Queue library used in the peak detection algorithm data processing

## **Windows Interface App (AcatActuator) Code Overview**

Program.cs – Main launching point of the application

AcatActuator.cs – Initializes forms and serial connection to Arduino

Form1.cs – Code for main window

FormFeedback.cs – Code for feedback window

SerialComm.cs – Establishes serial connection to Arduino and sends ACAT trigger events

Chart.cs – Chart component in main form

SensorPositionTracker.cs – Tracks the sensor positon when there is no movement and gives recommendations on sensor placement

WindowOverlapWatchdog.cs – Makes sure feedback window goes on top of all other windows except ACAT ones

Settings.cs – Keeps track of Arduino and application settings loaded from App\_Settings.xml

## **Licensing Information**

This project is distributed under the Apache License, Version 2.0.