## How to Start

#### IR Sensor

1. First, prepare the needed files. And connect IR sensor to Arduino, and Arduino to computer.
2. Upload the Arduino side program ‘readSharpSensor\_ros\_highFrequency.ino’ to arduino. This will need an Arduino IDE, which can be download at ‘https://www.arduino.cc/en/Main/Software’. It is possible that you may meet some error about ‘permission denied’. If this occurs, open a terminal and run:

$ sudo chmod 777 /dev/ttyACM0

‘ACM0' is the serial port of your Arduino, check it at your Aduino IDE in ‘tools’

Also please notice that the information output pin of the sensor should be connect to ‘A0’ port of Arduino. Otherwise, please change the ‘analog\_pin’ in the code.

1. Copy the ROS program to your ROS work space and run ‘catkin make’.
2. Start roscore, run:

$ roscore

1. Open a new terminal, run:

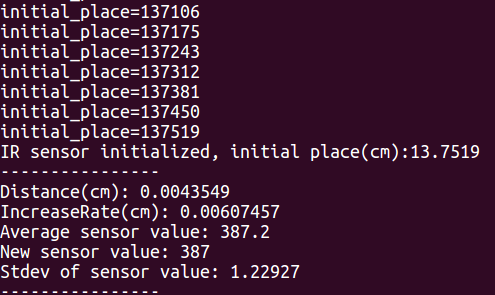
$ rosrun rosserial\_python serial\_node.py \_port:=/dev/ttyACM0

Similarly, ‘ACM0' is the serial port of your Arduino, check it at your Aduino IDE in ‘tools’

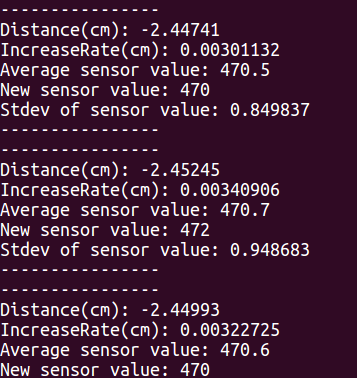
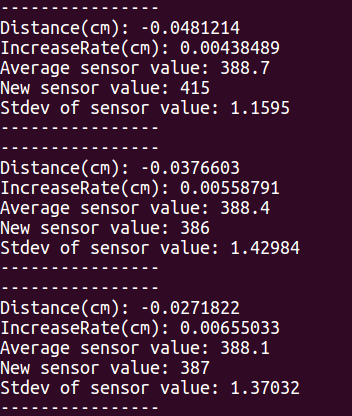
1. Open a new terminal, run:

$ rosrun AutoCircle\_generater IR\_Sensor

And it will immediately begin initialization. Make sure not to change the distance while initialization. The updating rate of this program is around 400 to 500 hz. When initialization is done, you could see:



Then, the terminal will constantly show the distance and some relative values. And the distance will be the relative distance about the initial place. Your could move the distance to check if the program is working well.



1. When this IR\_Sensor is running, it will provide a ros node ‘IR\_Sensor’, which has a service called ‘get\_IR\_distance’. This could be used to communicate with RAVEN program. There is an easy example called ‘test\_IR\_client’ in the codes. You could test it. Run:

$ rosrun AutoCircle\_generater test\_IR\_client

#### RAVENState\_Recorder

This is a recorder which could record the desired RAVEN state to a desired .txt file by just include a header file and add a line in the callback function. Compared with using rosbag, this program could generate a smaller recorder file and it is easy to be read by MATLAB.

To use this, copy ‘RavenState\_Recorder.cpp’ and ‘RavenState\_Recorder.h’ to the Autocircle code source file folder. Please read the instruction and the ‘define’ part of the header file. There are some changes which need to be done in ‘CMakeLists.txt’

In ‘Raven\_Controller.cpp’ file:

#include “RavenState\_Recorder.”

And in the callback function:

void Raven\_Controller::callback\_raven\_state(raven\_2::raven\_state msg),

Add:

recorder(msg)

Make sure it is outside the for loop.

### Technical Details

#### Initialization

The IR\_Sensor program has an initialization process. It will (currently) use the first 2000 sensor values to make sure the initial distance of the sensor and RAVEN tip. After this, the distance given by the IR\_Sensor program will be the relative distance about the initial place. This is to avoid transformation between absolute distance when this program is co-operating with RAVEN program.

#### Curve Fitting

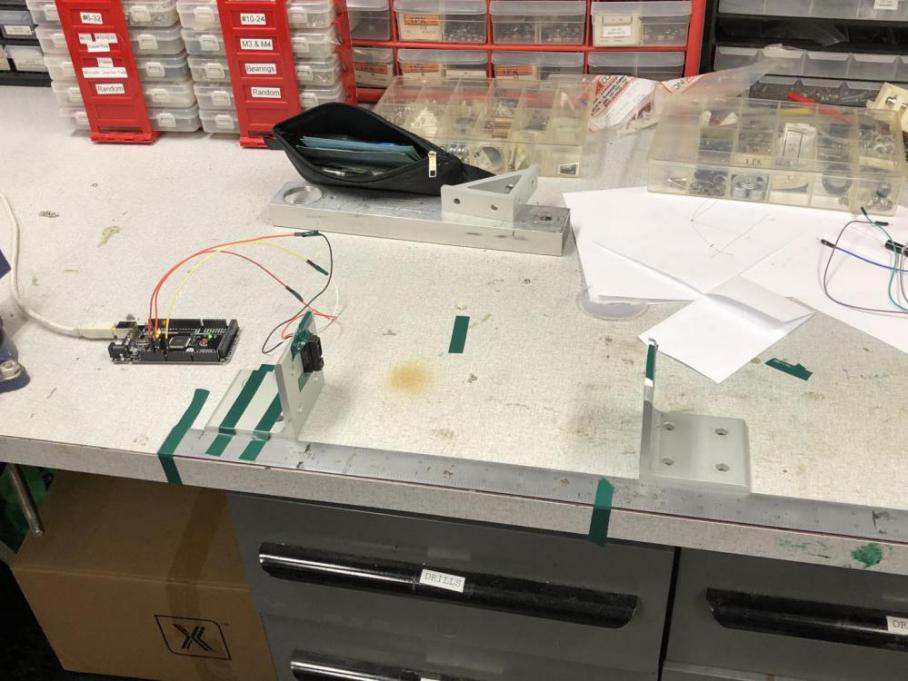
After I got this IR Sensor, I find that the relation of the sensor value and the distance given by the manual is not very accurate. Thus I borrow a nice and accurate ruler from ME department and record over 40 groups of values, where each group has over 200 values and the average of the values, which were in the 3 times of the standard deviation, were calculated.

At first, I tried to do curve fit over 10cm to 70cm, and I find that it will need at least 11th-order to get an acceptable result, which makes the parameters becomes very big or small. So I turn to use a low-order curve fitting. Thus a 4th-order version is created, however it is only valid for the range 10cm to 20cm, the most sensitive range of this sensor.









#### Filter

Because the IR\_Sensor is not very accurate and is very noisy. Thus a filter is needed. In order to make the filter real-time, at first, a simple moving average filter is used. This is by creating a vector which has a constant size of 10 to store the 10 newest sensor values. Every time updating the distance, the program will use the average of these 10 values instead of only the newest one. After implementing this, the performance gets better. But the result still shows relatively big inaccuracy, because the noise, I think, is not white in a short time period (although it is white when we get thousands of samples). Thus, I use another method to deal with this - neutralizing the samples that are 2 standard deviation away from the sensor value vector.

If the new sensor value is 2 standard deviation away from the sensor value vector. It will be replace by a weighted average of itself and the average of the 10 values. The weight is calculated by comparing with the double standard deviation.

long double dev\_para = abs(sensorval\_origin-sen\_val\_avr)/2\*sen\_val\_stdev;

sensorval\_filtered = (dev\_para\*sen\_val\_avr+sensorval\_origin)/(1+dev\_para);

After this, the result becomes stable and accurate. However, this filter has a delay. After some test, I found that for 1cm sudden change, the delay time is about 0.09s. And for 4cm change, the delay is about 0.4s. As the system will not change at this fast speed, I think this delay is acceptable.

#### Communication

**Sensor to IR Sensor Program:**

The IR sensor is connected with the computer using Arduino, by uploading a program to Arduino. It can send sensor signals through USB serial port to a ROS topic. Thus the program subscribes this topic to get the newest sensor signal. The updating rate is about 400 to 500 hz. However, normally, the rate of the Arduino ROS rate is around 120hz. The method to achieving this high frequency is to change the message type to a small one ‘int16’ (a lot of ROS message types have time stamp header, however this will significantly slow the rate down). It is not efficient to pass ‘double’ type. So passing ‘int’ sensor value and transferring it to the distance in IR Sensor Program should be a better choice.

Thus, the IR Sensor is relatively independent, it subscribes to a ROS topic generated by Arduino and provide a service to give out the distance.

**IR Sensor Program to Autocircle Program:**

IR Sensor Program is independent from Autocircle Program. So it will not affect Autocircle’s muti-thread console. The IR Sensor Program provide a ROS service server which can give the newest filtered distance. So the Autocircle Program just need to create a ROS service client to call the service.