FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Autonomous Intelligent Systems

Group – S4

RP-US-C Sensor Application for Autonomous Robots

Submitted By:	**Matriculation Number:**
Mir Mehedi Hasan Rayhan	1324345
Ashraf Uz Zaman	1324581

Examiner: Peter Nauth

Date of Submission: 14th May 2021

Contents

1. Abstract:

Navigation is a critical capability for mobile robots, and autonomous navigation has made significant progress in the past. Most navigation systems built so far, on the other hand, are focused on indoor navigation, navigation through rugged outdoor terrain, or navigation based on road use. Robots that can effectively maneuver through urban environments and public zones must overcome several obstacles, particularly diverse three-dimensional configurations and rapidly evolving scenes, as well as inaccurate GPS data.

The aim is to create software that can receive, view, and save ultrasonic signals from the Red Pitaya using a Raspberry Pi. The aim is to keep the system stable. The first step is to collect data from a 1-meter-range ultrasonic sensor. Using an external trigger condition, capture the time when object data is available to obtain the data with the lowest noise ratio.

Before working on the feature component, we cut down the input part to make it more precise so that we can get the desired and reliable data for each object.

Key Features:

- High performance hardware
- Replaces most essential instruments like Oscilloscope, Spectrum analyzer, Signal generator, LCR meter
- LAN or wireless access from any WEB browser via tablet or a PC regardless of the OS (MAC, Linux, Windows, Android, iOS) LabVIEW and MATLAB® interface.
- Possibility to make your own application and share it with others
- Open-source software

2. Principal of Project:

In this project we used RP-US-C sensor for robots so that it can differentiate between human and object. If human is in front of the robot, the sensor will indicate light and if there are objects in front of the sensor, the sensor will show the different light: That´s how the sensor will detect whether it is human or object. Here we used C programming Language to run the sensor. We used fft, psd, and variance to find features, and then we applied machine learning, specifically Nave Bayes, to the extracted features and trained our method.

The Raspberry Pi is a low-cost device that connects to a computer monitor or television and operates using a traditional keyboard and mouse. It allows us to create our own applications and program in languages such as Scratch and C programming. It can perform all of the tasks that a desktop computer can, including searching the internet and writing software.

The aim of using this is to be able to connect it with Red Pitaya to demonstrate how one device connected to the sensor can send data and how we can use the data.

3. Elements and Technology:

For achieving this we are using multiple technology and entities. That has been described here.

3.1 RedPitaya:

RedPitaya is a piece of open-source hardware that can be used to replace a variety of costly laboratory measurement and control instruments. Red Pitaya provides open-source controlling and monitoring software that includes simple visual programming software and free, fully prepared open-source, web-based analysis and calculation equipment that operate on solid, credit card-sized boards. The board can be turned into a web-based oscilloscope, spectrum analyzer, signal generator, LCR meter, Bode analyzer, and several other programs with a single tap. Matlab, LabVIEW, C# and Scilab can all be used to manipulate RedPitaya.For our software creation, we use C programming.

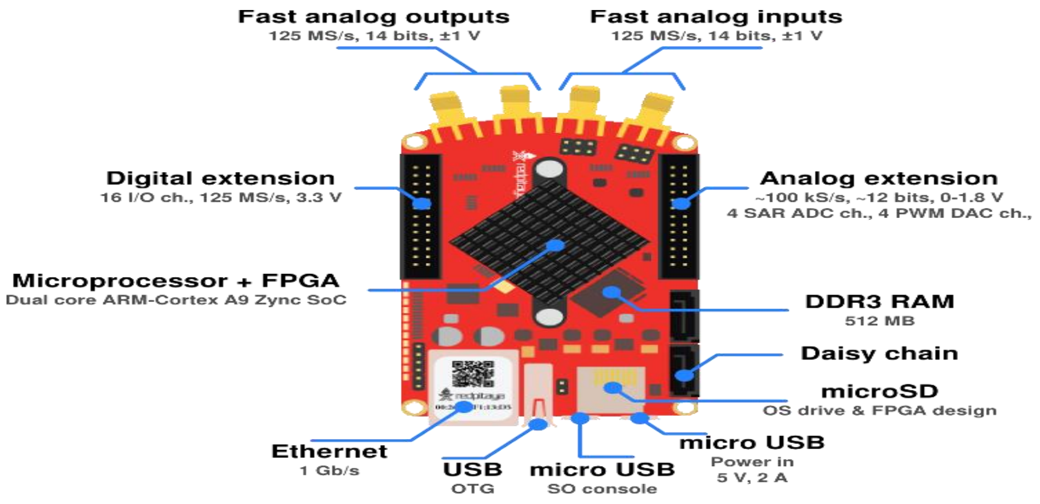


Fig: Architecture of RedPitaya

3.2 Support Vector Machine (SVM):

For two-group classification problems, a support vector machine (SVM) is a machine learning algorithm concept that combines classification algorithms. SVM models will categorize new text after being given sets of labelled training data for each group.

Advantages of support vector machines:

- In high-dimensional spaces, it works well.
- When the number of dimensions exceeds the number of observations, the method is still accurate.
- It is memory effective since it uses a subclass of training points (called support vectors) in the decision function.
- Different Kernel functions can be defined for the decision function, making it versatile. Common kernels are included, but custom kernels can also be specified.

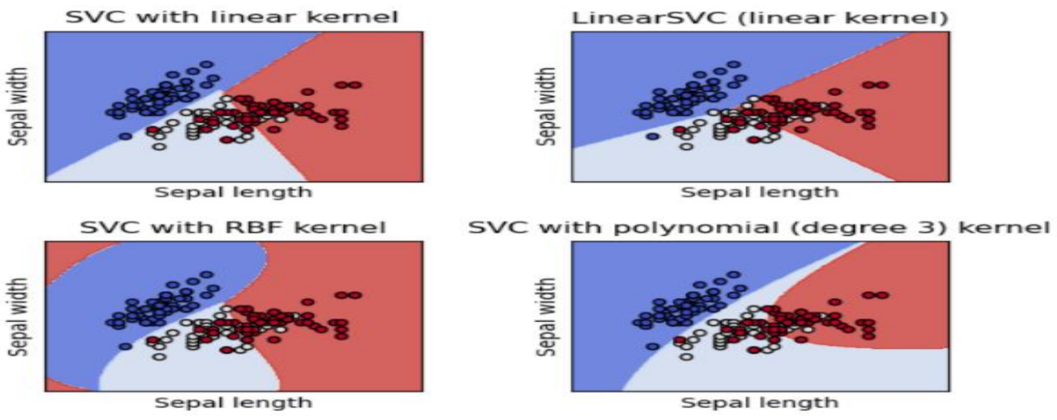
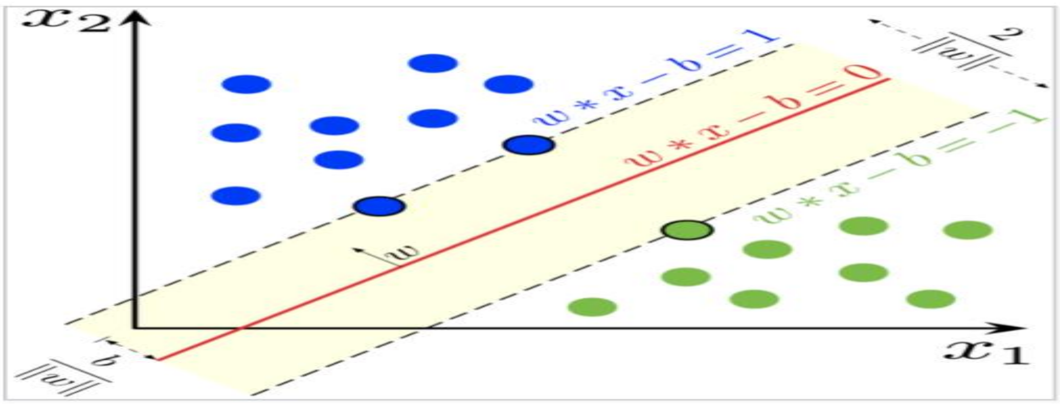


Fig: Types of SVM

A simple binary classifier, the SVM classifier is often talked about during Machine Learning interviews.

We concentrate on linear SVM on binary classification with gradient descent optimization approach for simplicity and interview planning.

Problem: using a dataset with two input dimensions, X1 and X2, classify two groups in Y.



We try to fit a linear line that separates the two groups and maximizes the distances between them.

The line is represented with

w*x + b = 0

For label 1, w*x + b ≥ 1

For label -1, w*x + b ≤ -1

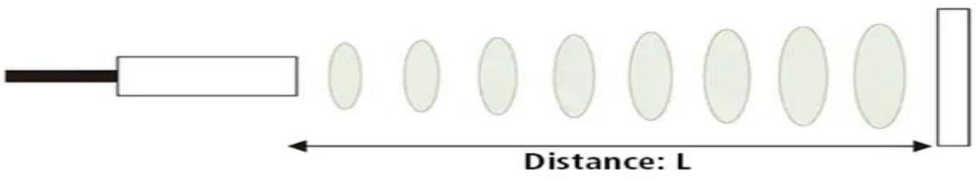
The distances of the gap in-between two classes 2/|w|

This form of SVM cost function is called Hard-Margin cost function when we try to find a line that perfectly divides the two groups.

We may use Hinge-Loss when designing the cost function if any of the outliers are misclassified.

3.3 Ultrasonic Sensor:

Ultrasonic devices, as the name implies, use ultrasonic waves to measure distance. The sensor head sends out an ultrasonic wave, which is reflected back to it by the target. Ultrasonic sensors use the time between emission and reception to calculate the distance to the target.



A transmitter and receiver are used in an optical sensor, while an ultrasonic sensor uses a single ultrasonic element for both emission and reception. A single oscillator emits and receives ultrasonic waves alternately in a reflective model ultrasonic sensor. The sensor head can be made smaller as a result of this.

3.4 Data Structure:

A data structure is a standardized file containing, sorting, and analyzing data. There are a variety of simple and specialized data structures available, all of which are configured to organize data for a particular purpose. Data systems sum it up for customer to access and interact with the information they require. Most significantly, data structures define how knowledge is organized so that computers and users may understand it properly. A data structure could be chosen or structured to keep records for use with different algorithms in computer and data engineering. In certain cases, the basic operations of the algorithm are inextricably linked to the data structure's architecture. Each data structure includes details about the information elements, interrelationships, and, in certain scenarios, data-processing functions.

The data structure and its related processes for example, are linked together as sort of a group structure in an object-oriented programming language. There may be functions defined to work with the data structure in non-object-oriented languages, but they are not practically part of the information system.

3.5 Absolute Energy of the Signal

The sum of the squared values of the time signal is the signal's absolute energy. The following is the mathematical equation for Absolute Energy over "x":

$$E = \sum_{i=o,1,\ldots,n} x_i^2$$

The time series signal to measure the function of is defined by the parameter "x." This will return the value of this function. This feature's return form will be "double."

4. Project Briefing:

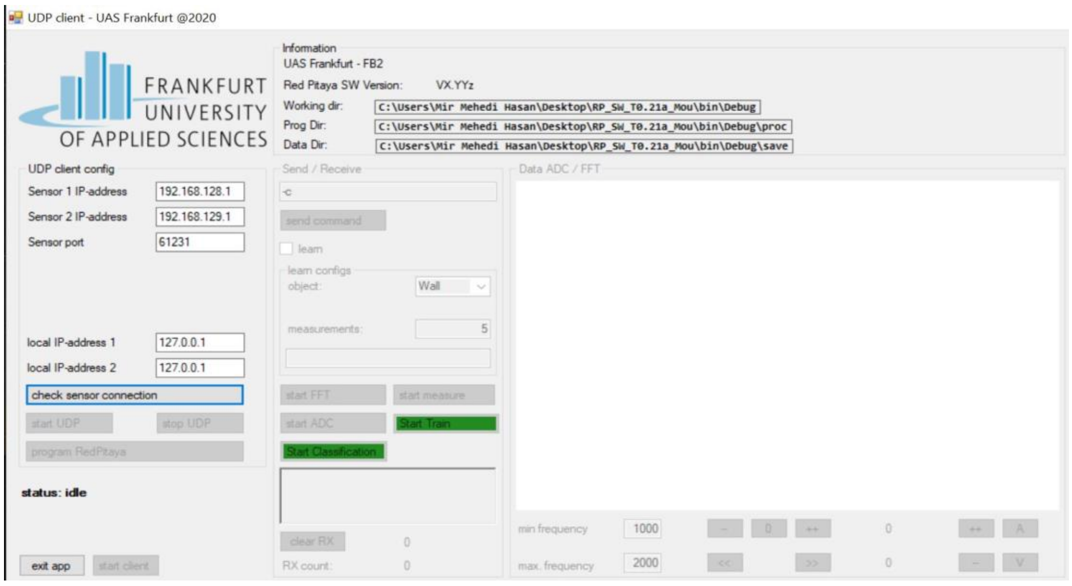
In this project we took data for human and objects for classification. For this we used the following steps:

1. Establish communication between GUI and Ultrasonic sensor
2. Collecting human and wall data using ultrasound sensor
3. Train SVM

4. Testing the real time classification using trained SVM
5. Showing the output result using blinking LED

5. Implementation and Demonstration:

- GUI interface which is done by C#



UDP client - UAS Frankfurt @2020

Information
UAS Frankfurt - FB2
Red Pitaya SW Version: VX.YYz
Working dir: C:\Users\Mir Mehedi Hasan\Desktop\RP_SW_T0.21a_Mou\bin\Debug
Prog Dir: C:\Users\Mir Mehedi Hasan\Desktop\RP_SW_T0.21a_Mou\bin\Debug\proc
Data Dir: C:\Users\Mir Mehedi Hasan\Desktop\RP_SW_T0.21a_Mou\bin\Debug\save

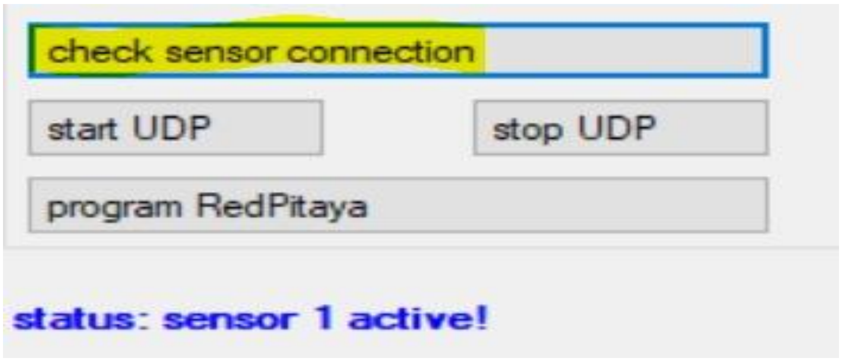
UDP client config
Sensor 1 IP-address 192.168.128.1
Sensor 2 IP-address 192.168.129.1
Sensor port 61231
local IP-address 1 127.0.0.1
local IP-address 2 127.0.0.1

status: idle

Send / Receive
-c
learn configs
object: Wall
measurements: 5
RX count: 0

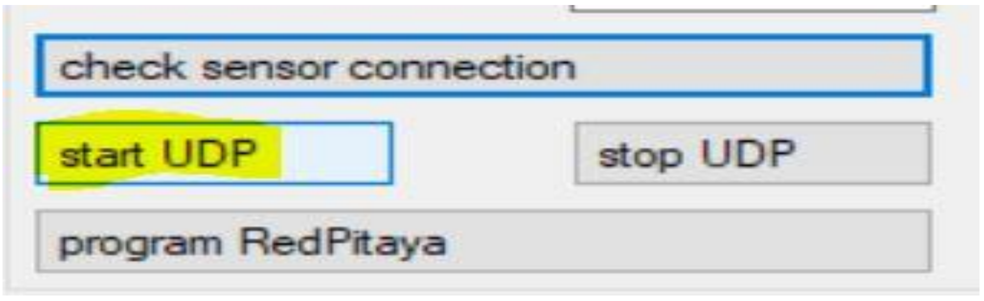
Data ADC / FFT
min frequency 1000
max. frequency 2000

- Here we check the connection

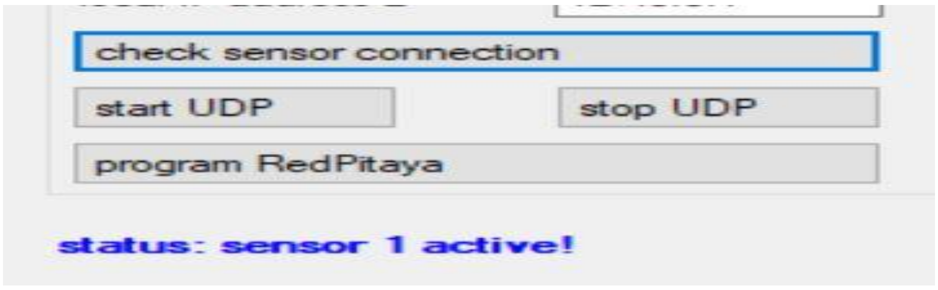


status: sensor 1 active!

- Here we start the UDP

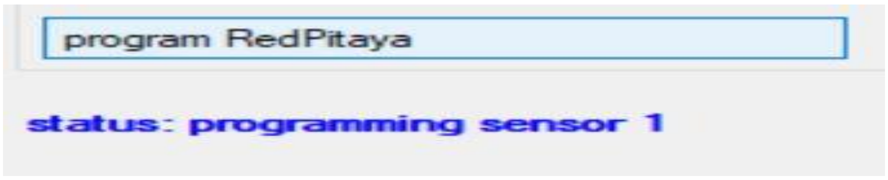


- Activation of sensor 1



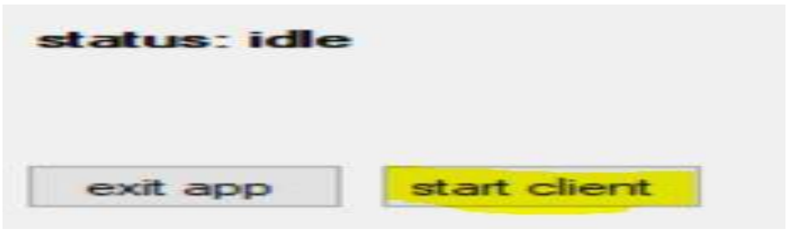
status: sensor 1 active!

- Programming RedPitiya



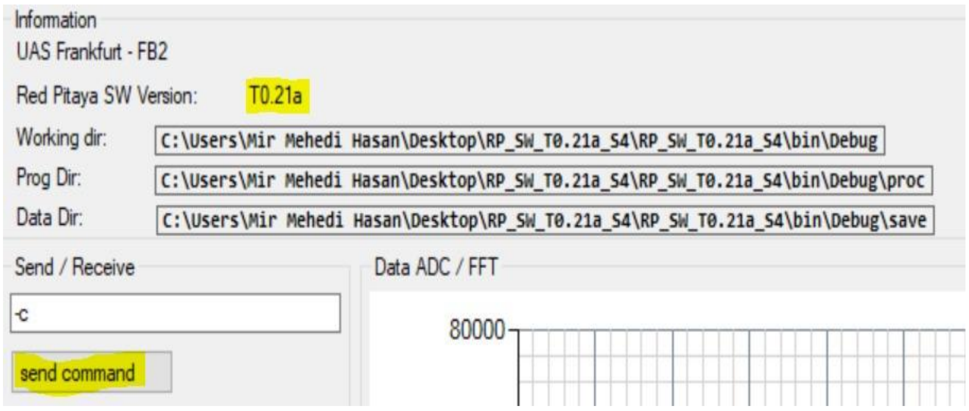
status: programming sensor 1

- After successfully program we will start the clint



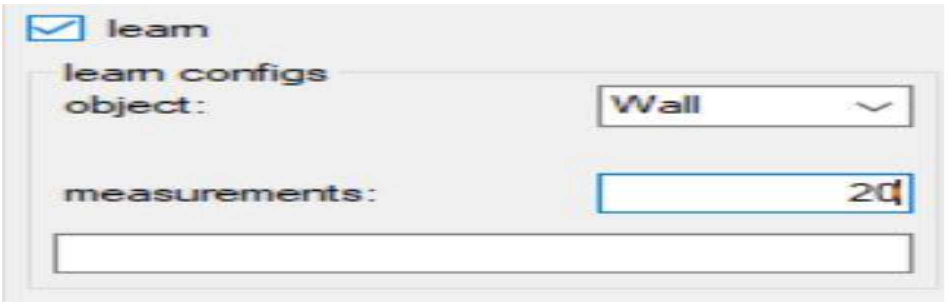
status: idle

- Checking the sensor status



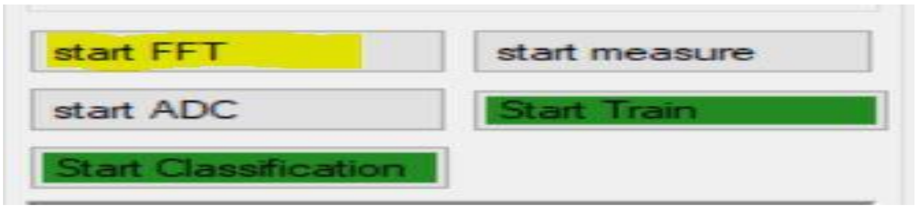
UAS Frankfurt - FB2
Red Pitaya SW Version: T0.21a
Working dir: C:\Users\Mir Mehedi Hasan\Desktop\RP_SW_T0.21a_S4\RP_SW_T0.21a_S4\bin\Debug
Prog Dir: C:\Users\Mir Mehedi Hasan\Desktop\RP_SW_T0.21a_S4\RP_SW_T0.21a_S4\bin\Debug\proc
Data Dir: C:\Users\Mir Mehedi Hasan\Desktop\RP_SW_T0.21a_S4\RP_SW_T0.21a_S4\bin\Debug\save
-c

- Taking sample data for the objects

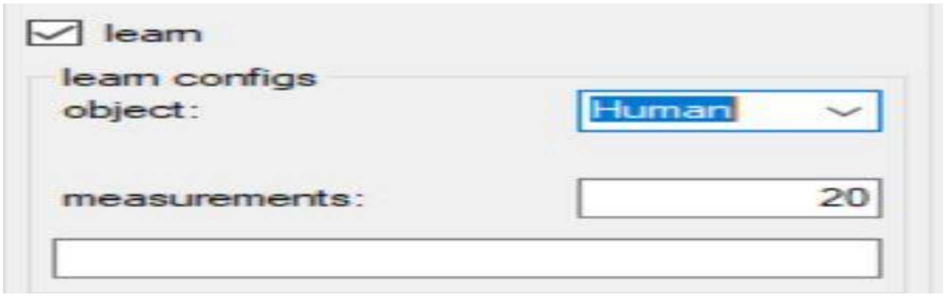


object: Wall
measurements: 20

- Start FFT for taking sample data

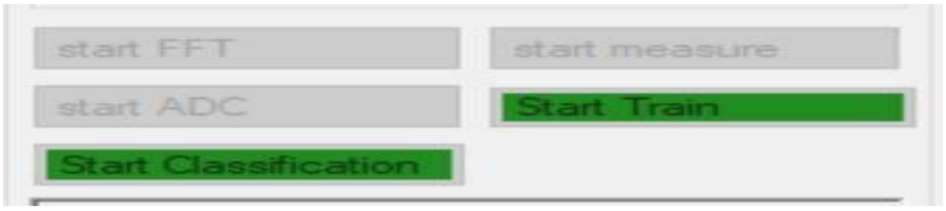


- Taking sample data for Human

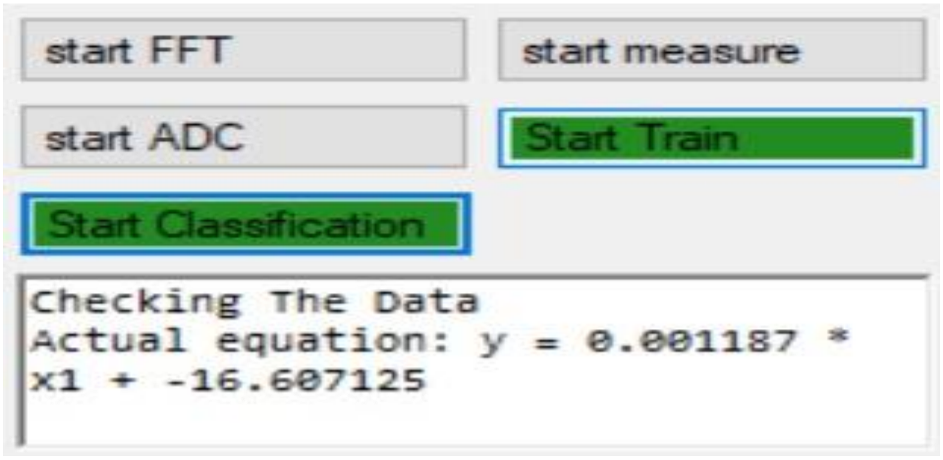


learn
learn configs
object: Human
measurements: 20

- Train the SVM then compare with real data

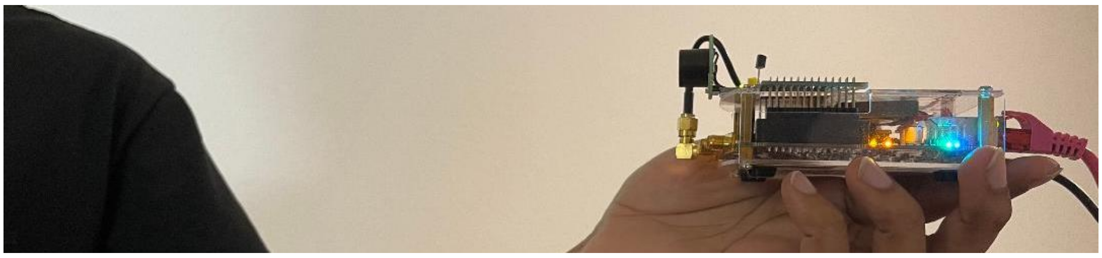


- Ready for real time classification



```
Checking The Data
Actual equation: y = 0.001187 *
x1 + -16.607125
```

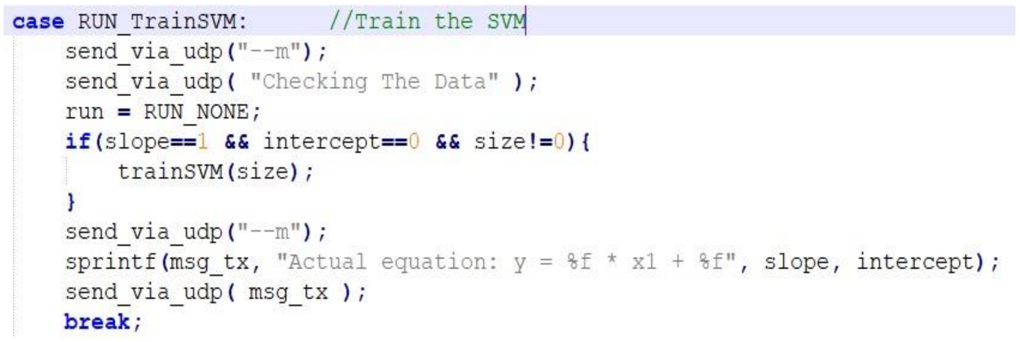
- For human first LED is blinking



- For Object Second LED is blinking

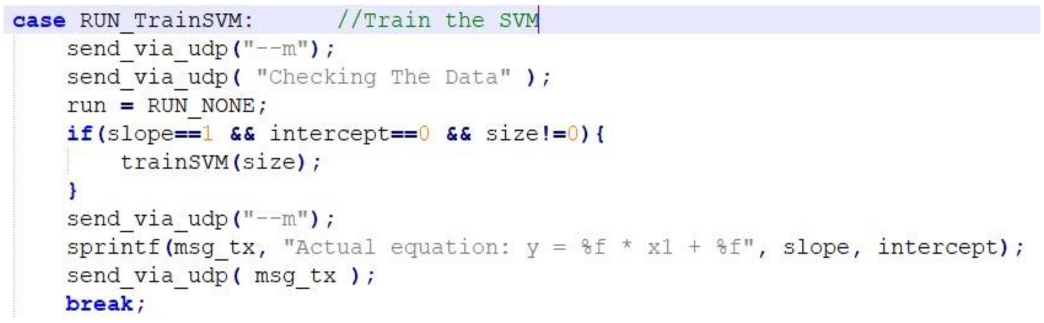
6. C Programming Code Details:

- Here we call the train function



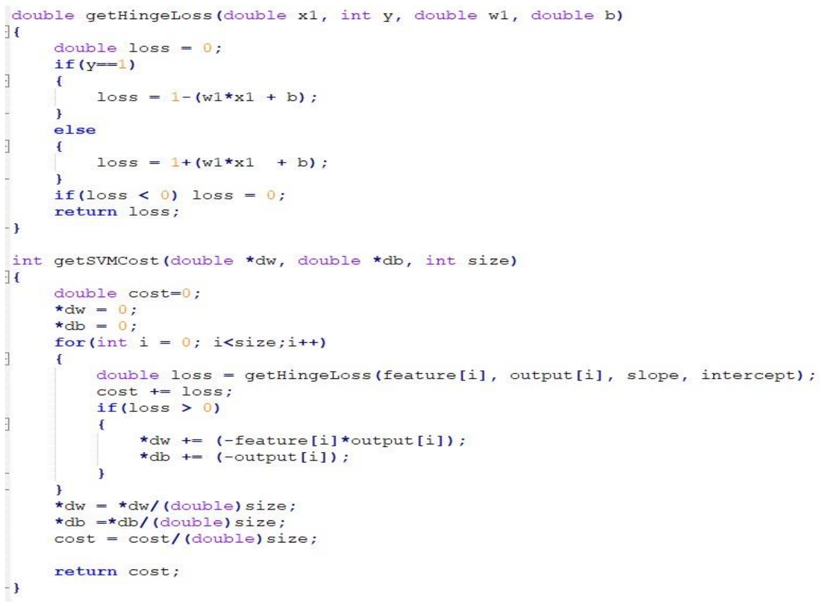
```c
case RUN_TrainSVM:        //Train the SVM
    send_via_udp("--m");
    send_via_udp( "Checking The Data" );
    run = RUN_NONE;
    if(slope==1 && intercept==0 && size!=0){
        trainSVM(size);
    }
    send_via_udp("--m");
    sprintf(msg_tx, "Actual equation: y = %f * x1 + %f", slope, intercept);
    send_via_udp( msg_tx );
    break;
```

- SVM train function



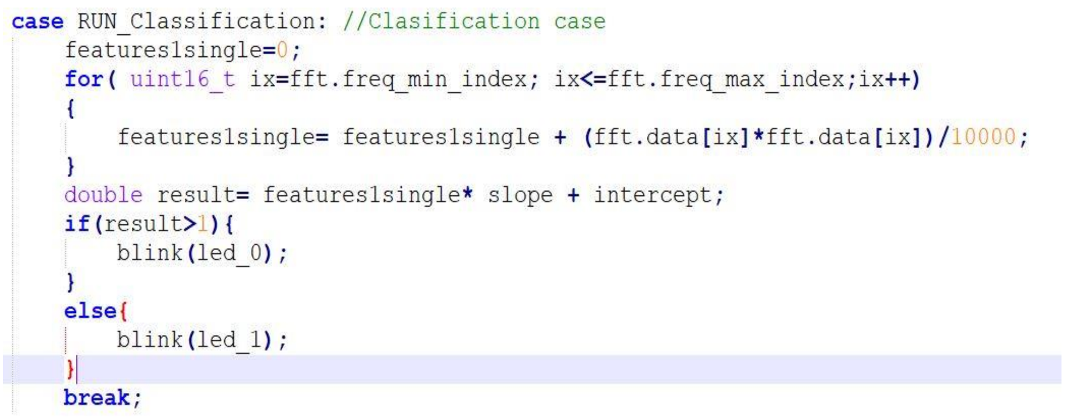
```c
case RUN_TrainSVM:        //Train the SVM
    send_via_udp("--m");
    send_via_udp( "Checking The Data" );
    run = RUN_NONE;
    if(slope==1 && intercept==0 && size!=0){
        trainSVM(size);
    }
    send_via_udp("--m");
    sprintf(msg_tx, "Actual equation: y = %f * x1 + %f", slope, intercept);
    send_via_udp( msg_tx );
    break;
```

- Feature extraction for training the SVM



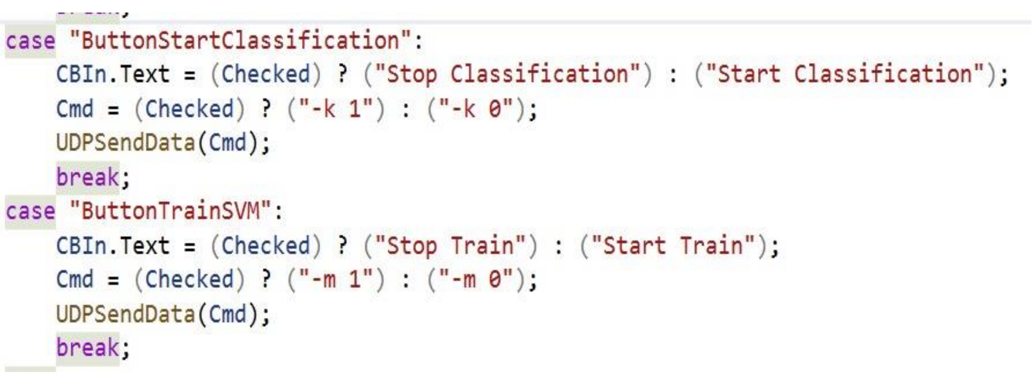
```c
double getHingeLoss(double x1, int y, double w1, double b)
{
    double loss = 0;
    if(y==1)
    {
        loss = 1-(w1*x1 + b);
    }
    else
    {
        loss = 1+(w1*x1  + b);
    }
    if(loss < 0) loss = 0;
    return loss;
}

int getSVMCost(double *dw, double *db, int size)
{
    double cost=0;
    *dw = 0;
    *db = 0;
    for(int i = 0; i<size;i++)
    {
        double loss = getHingeLoss(feature[i], output[i], slope, intercept);
        cost += loss;
        if(loss > 0)
        {
            *dw += (-feature[i]*output[i]);
            *db += (-output[i]);
        }
    }
    *dw = *dw/(double)size;
    *db =*db/(double)size;
    cost = cost/(double)size;

    return cost;
}
```

- Realtime Classification



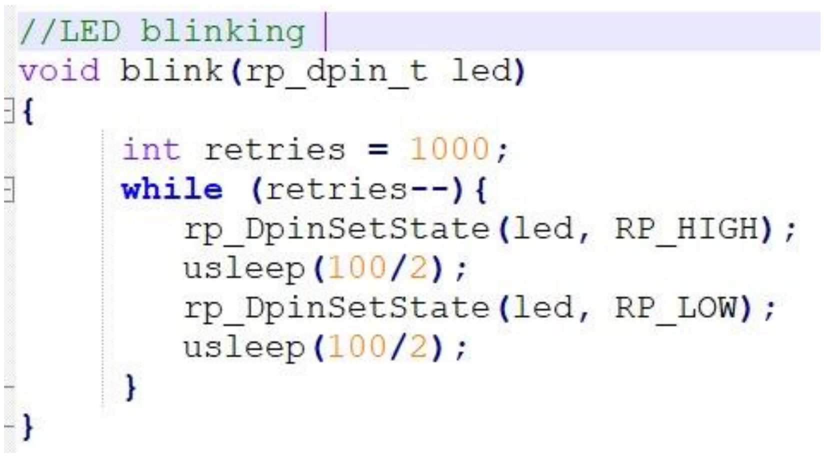
```
case RUN_Classification: //Clasification case
    features1single=0;
    for( uint16_t ix=fft.freq_min_index; ix<=fft.freq_max_index;ix++)
    {
        features1single= features1single + (fft.data[ix]*fft.data[ix])/10000;
    }
    double result= features1single* slope + intercept;
    if(result>1){
        blink(led_0);
    }
    else{
        blink(led_1);
    }
    break;
```

- C code for customized button.



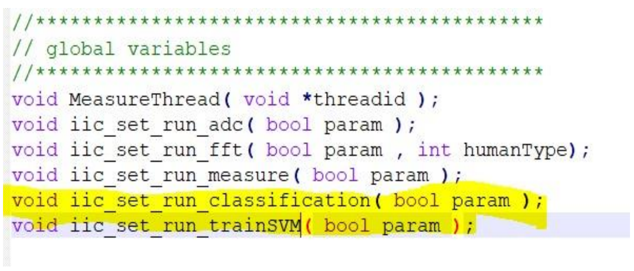
```
case "ButtonStartClassification":
    CBIn.Text = (Checked) ? ("Stop Classification") : ("Start Classification");
    Cmd = (Checked) ? ("-k 1") : ("-k 0");
    UDPSendData(Cmd);
    break;
case "ButtonTrainSVM":
    CBIn.Text = (Checked) ? ("Stop Train") : ("Start Train");
    Cmd = (Checked) ? ("-m 1") : ("-m 0");
    UDPSendData(Cmd);
    break;
```

- Code for LED blinking



```
//LED blinking
void blink(rp_dpin_t led)
{
    int retries = 1000;
    while (retries--){
        rp_DpinSetState(led, RP_HIGH);
        usleep(100/2);
        rp_DpinSetState(led, RP_LOW);
        usleep(100/2);
    }
}
```

- Defined the global variable



```
//*********************************************
// global variables
//*********************************************
void MeasureThread( void *threadid );
void iic_set_run_adc( bool param );
void iic_set_run_fft( bool param , int humanType);
void iic_set_run_measure( bool param );
void iic_set_run_classification( bool param );
void iic_set_run_trainSVM( bool param );
```

7. Discussion and Problems:

While implementing the given task there are some issues arises.

a. Ultrasound sensor is very sensitive with sound. So, if the data taken in a crowded place, it does not show the result properly.
b. The distance between the sensor face and the object is very important. Because ultrasonic sensor cannot generate a sound wave signal and pick-up a signal returning from a target at the same time. So, an inherent dead-zone beginning from the face of the sensor and extending outward.
c. Facing angle between sensor and the object is very important, while taking the data.
d. Having difficulties on implementation of classification program in C because of the limitation of resources.

For solving all of this we go through the internet, reading blogs. It takes more time of our project to cater these problems. Meanwhile, we are in contact with other groups they also have common issues.

8. Result:

In this project we saw that when any object comes in front of the sensor First LED start blinking and when human comes in front of the sensor the second LED start blinking.

9. Improvements and Implementation:

Our goal is to improve the C library for using the RedPitaya device which is using for sensing the human and the object.

10. Conclusion:

The aim of this project was to detect human or object. Here we have connected the sensor with the laptop using wifi. The sensor has collected data for human and objects using ultrasonic sound sensor. Then the result was tested using trained SVM and the result was shown by the blinking of LED.

11. References

[1] Russsun, “https://medium.com/,” medium.com, 09 April 2019. [Online]. Available: https://medium.com/@dr.sunhongyu/machine-learning-c-svm-support-vector-machine-simple-example-deff5d55d43e. [Accessed 14 May 2021].

[2] “ Dexter Industries., “Five Ways To Run a Program On Your Raspberry Pi At Startup” (references).”.

[3] “ Red Pitaya d.d. Revision a25e54a3, “STEMlab boards comparison.” 2017 (references).”.

[4] “ Trenz Electronic , “Red Pitaya Product” (references)”.

[5] RedPitaya, “Blink,” RedPitaya, 03 02 2021. [Online]. Available: https://redpitaya.readthedocs.io/en/latest/appsFeatures/examples/dig-exm1.html#code-c. [Accessed 14 05 2021].