Project Based Learning:

Insert Catchy Project Title

# Project Overview:

This App as an extension of the Android feature “smart lock” adding on an easy controllable proximity security feature, the user can set an perimeter where the device would only able to work with in the proximity of the trusted device but wound lose access when out of the perimeter.

# Phases:

Breaking down of the objectives for each phase.

* Building app
* Be able to read RSSI values form a BLE device
* Be able to lock the phone on command
* Be able to read RSSI values in the background
* Gather RSSI value
* Get RSSI value and the associated distance
* Analyzing the data collected
* Determent the Relationship between RSSI value and distance
* Use the data to run analysis with (Linear, polynomial, exponential regressions)
* Apply analysis result to app
* Pick the best regression and apply that to the app’s algorithm.

Building app

# Required Hardware:

* A Bluetooth low Energy device (trusted device)
* An Android phone (pixel 3, API 9)

# Objectives:

* Be able to read RSSI values form a BLE device
* Be able to lock the phone on command
* Be able to read RSSI values in the background

# App Functions:

* Admin access for the device
* The admin access is important for the power off function, also makes the app an admin app which be able to run in the background. To ask for Admin access go into Admin class. For more detail (2)
* Ask for necessary permissions
* In Android’s SDK 9 and above it is required to ask for permission such as location, Bluetooth, Foreground service etc. without those permission nothing on this app wound work, this is all done in the AndroidManifest.xml. (3)
* Run background task
* After asking for required permission you need to create a foreground service and add it in Manifest.xml. Note this foreground service must be on a different thread than the app, or else service will end when app is quit.
* Read RSSI value
* Reading RSSI value is one of the most important part with this project, it allows us to use it as a measure for distance. I have tried using Blutooth.gattCallBack() this doesn’t work with the SDK I was working with, which was SDK10. Also tried using Bluetooth.RSSI.callback() class, this works but it read RSSI at a speed of (0.2read/sec) which is not frequent enough for our purposes.
* The only one that worked was BLE.scanCallBack() , it is fast and reliable, and it is whar most people use to build RSSI related apps.(1)
* Store data in a SQL database and accessing it
* The data storing is for the testing phase, we Record the RSSI value and measure the distance that goes with it to use later for analysis.
* For SQL it is new to me, so I watched a video on YouTube (4). This guys video is all I needed for my purpose. It also tells you where to get the stored data, then all you need is a SQLite app to open it up and within the app you can convert it to a CSV file for analysis.
* I recorded the RSSI and Time, distance away from the trusted device.
* Show live chart of RSSI value update
* “MPAndroidChart” is what I used to create the live chart in our app, Talkie’s(5) video was a big help for me to get familiar with how this works.
* The X would be the time passed and the Y would be the RSSI value
* The purpose of this chart is to see visualized the effect distance have on the RSSI reading, it shows the farther the distance the lower the RSSI.
* Bluetooth functions
* The class provide a list of helpful functions such as checking if device is connected and available Bluetooth device nearby.
* Background running service
* ForegroundService is what I used for the reading of RSSI to run even when the App is closed
* Understand the tasks for different broadcast receivers

Gathering Data

# Objectives:

* Get RSSI value and store it in CSV file for analysis

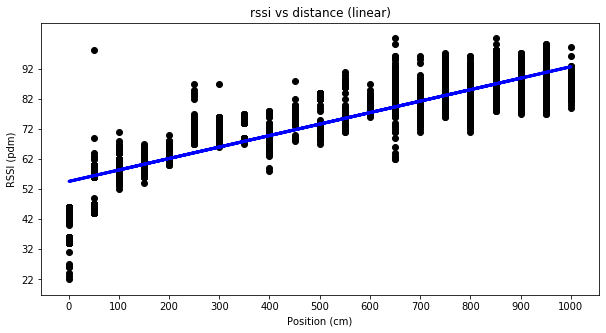
|  |  |  |  |
| --- | --- | --- | --- |
| index | time(sec) | RSSI(pdm) | Degree |
| 1 | 0.92 | -21 | 0 |
| 2 | 1.73 | -22 | 0 |
| 3 | 3.38 | -21 | 0 |
| 4 | 4.42 | -23 | 0 |
| 5 | 7.06 | -24 | 0 |
| 6 | 9.86 | -19 | 0 |

Analyzing

# Objectives:

* Use the data collected to find what is the relationship between distance and RSSI reading

# Objectives:



# Function

Y = 0.0381x + 54.23

# Mean squared error using cross validation (5 folds)

[46.64793598, 40.88163413, 38.84761443, 39.5697662, 36.24255513]

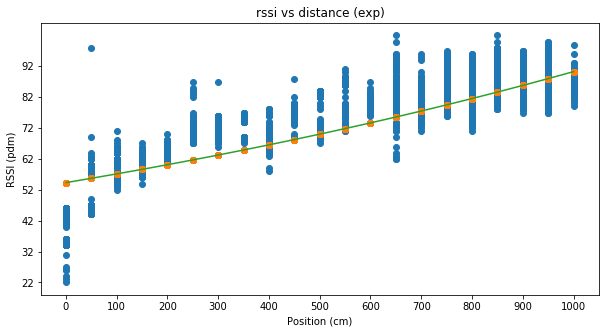
# Coefficient of determination using cross validation (5 folds):

[0.7878381, 0.78218156, 0.7801198, 0.77848775, 0.78601701]

# Over all

Mean squared error: 40.44

Coefficient of determination: 0.78



# Function

Y = 7.36897779ln(x) + 32.57858381

# Mean squared error using cross validation (5 folds)

[45.58121831, 47.65388994, 42.69361227, 47.72643158, 41.63482752]

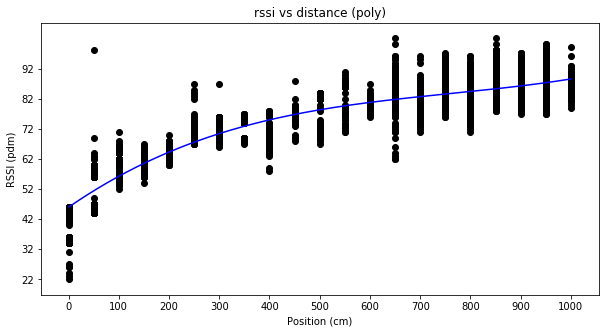
# Coefficient of determination using cross validation (5 folds):

[0.75459131, 0.75991652, 0.75388746, 0.75777903, 0.76308459]

# Over all

Mean squared error: 45.06

Coefficient of determination: 0.758



# Function

Y =0.0820925017x - 0.0000533687837x^{2} + 0.0000000118062673x^{3} + 48.42

# Mean squared error using cross validation (5 folds)

[30.00080044, 26.79459867, 24.40611771, 24.38920825, 27.2873735]

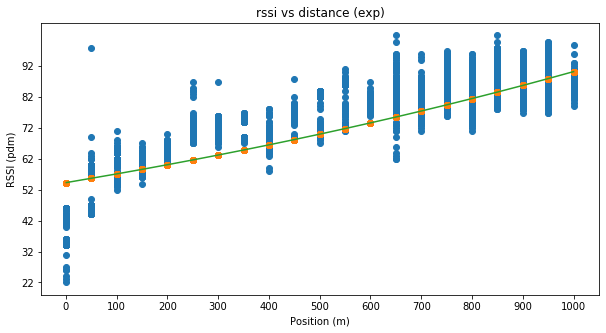
# Coefficient of determination using cross validation (5 folds):

[0.85953009, 0.85663014, 0.85385468, 0.85212652, 0.86387]

# Over all

Mean squared error: 26.57

Coefficient of determination: 0.86



# Function

Y = 43.9946 e^0.000508092x + 12.2

# Mean squared error using cross validation (5 folds)

[67.6303561, 52.2446, 50.1313561, 50.131356, 47.2873735]

# Coefficient of determination using cross validation (5 folds):

[0.709597, 0.7349403, 0.734940, 0.80212652, 0.711114]

# Over all

Mean squared error: 53.48

Coefficient of determination: 0.74

|  |  |  |
| --- | --- | --- |
| Regression models | Mean Squared error | Coefficient of determination |
| Linear | 40.44 | 0.78 |
| Polynomial | 26.57 | 0.86 |
| Exponential | 53.48 | 0.74 |
| logarithmic | 45.06 | 0.76 |

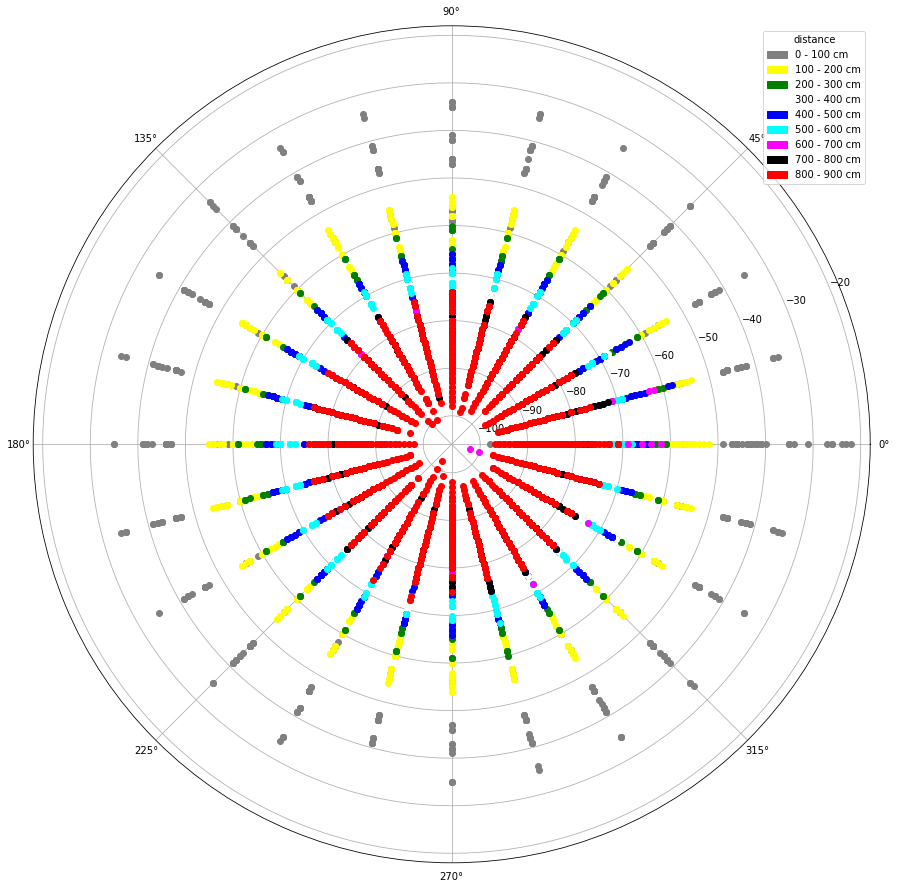
Best: Polynomial

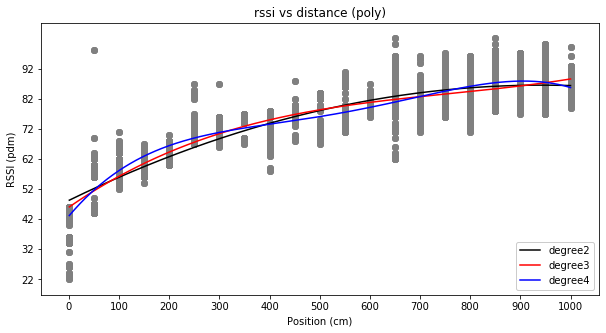
Worst: Exponential

Summary

The best fit is a 3rd  degree polynomial regression, with a mean Squared error of 26.57 and a coefficient of determination of 0.86. Above is a compare chart of all the regressions that’s been done in this experiment, the average is gained from the cross validation’s (5-fold), round up to the second decimal places.

Data used in the regression model is all the data points collected from degree 0 to 360 with 15 degrees increment and a 50 cm increment in distance below is a polar graph representing all of them.





|  |  |  |
| --- | --- | --- |
| Polynomial of degree | Mean Squared error | Coefficient of determination |
| 2 | 28.02 | 0.85 |
| 3 | 26.57 | 0.86 |
| 4 | 22.94 | 0.877 |

Even though polynomial of degree 4 got the best MSE and R^2 but it is not a one to one function therefore I choose the second best which is degree 3, also a one to one function.

# Notes:

It is important to use a BLE device because it has a faster RSSI reading, The RSSI value send off by a regular Bluetooth device is not frequent enough (0.2read /sec) for our purposes. With BLE we can get as frequent as(2read/sec), here I used a BLE library on git(1) that has a RSSI read function build in, also includes a lot of exception handling like device compatibility.

There is a lot of googling, at first glance a lot things seems impossible but after couple hours of struggling sometimes days you always find a way to do what you want to do. There are things I wish I did differently, it would of saved a lot of my time. Key notes, if you can do something yourself then don’t use others code, I ended up recoding a lot of the code I used from Github, so I understand it better and use it to my will.

# Summarize:

# Links resources

1. <https://github.com/alt236/Bluetooth-LE-Library---Android>
2. <https://developer.android.com/guide/topics/admin/device-admin>
3. <https://developer.android.com/guide/topics/manifest/permission-element>
4. <https://www.youtube.com/watch?v=hDSVInZ2JCs>
5. <https://www.youtube.com/watch?v=DD1CxoVONFE>