COLLEGE OF COMPUTING AND MATHEMATICS
Computer Engineering Department

COE 446 - Mobile Computing
Project Report
Project: Proximity-based Authentication

Team 2

St 1: Yusef Seto ID: 201963010

St 2: HASSAN AL NASSER ID: 201927570

St 3: ABDULMOHSEN ALANAZI ID: 201839080

Term 231

TABLE OF CONTENT

INTRODUCTION

Proximity-Based Authentication is a project proposed in the Mobile Computing course (COE 446), with the aim of designing a process that utilizes Radio Frequency (RF) signals to authenticate the identity of a digital device user based on one or multiple signal signatures and the user proximity to that device. Before continuing further, there are two main devices: receiver and transmitter, depending on the approach, the first could be carried by the user to authenticate himself/herself and the other might be the one that needs authentication and does so by receiving a legit intentional acknowledgment from the receiver and is connected to the device that needs authentication, more information is in *Design* and *Approaches* sections.

The requirements and deliverables of the project are as follows:

1. Collect and analyze signals and estimate the distance.
2. Provide information about the users, devices used and authentication status.

This report will analyze the problem further, provide the constraints and specifications that have been considered, and discuss the physical design and implementation of the approaches and testing procedure. Also, it will provide a description of the problems faced, lessons learned, and what could be added to improve the result. Finally, work distribution, conclusion, references, and appendixes are detailed.

PROBLEM STATEMENT

The problem can be broken into two main parts:

- The authenticity of the received signal by the device. The signal received to authenticate must be a unique and precise one that can be recognized by the transmitter.
- The proximity of the user to the device. The signal has been received from a legitimate distance from the device that needs authentication.

These are the main two issues that were under consideration while working on this project, each will be discussed thoroughly in further sections.

CONSTRAINS

Following the first requirement mentioned above, the projects must operate as close as possible to the physical layer. Thus, no usage of a link layer protocol is allowed, e.g. Wi-Fi or Bluetooth. Through the research phase, we found out that accessing the Network Interface Card (NIC) of both a phone and a windows computer to receive vanilla RF signals, do not follow any protocol, is not feasible. As a result, we had to find another adequate approach. Further discussion about the approach is provided in the next section.

SPECIFICATIONS

The focus throughout the period of the project is to find approaches to the next:

- Creating unique signals that can be sent securely from the user to be received correctly by the device.
- The device can figure the if the user is close to it with considerable accuracy, around 85 percent.
- The chosen procedures can be tested.
- The procedure can be simply integrated into an application, either web or mobile.

PHYSICAL DESIGN

Since we could not use a mobile and Personal Computer (PC), our physical design is as follows: we will use two Arduino boards, one will act as the receiver and the other as the transmitter, which will be connected to the device that needs authentication. The first will be a substitute for the user's mobile phone and receive the signal. The second will be the NIC of the transmitter as the collector of the signal and acknowledgments. As can be seen in Figure 1, the transmitter is connected to a computer using a wired serial interface, Universal Serial Bus (USB), and the orange circle is the area where the user can be authenticated.

The description of used components is as follows:

- The Arduino board that is used is NodeMCU 1.0[1], can be seen in Figure 2.
- The other component is two NRF24L01[2] transceivers which each will be connected to one Arduino board via Serial Peripheral Interface (SPI) protocol, can be seen in Figure 3. This transceiver utilizes the 2.4 GHz Industrial, Scientific, and Medical (ISM) bandwidth with many features:
    - Channel selection (frequency) from 125 channels
    - Adjustable Power Levels.
    - Utilizes Gaussian Frequency Shift Keying (GFSK)
    - And many other can be seen in the component documentation. [2]

  At first, we tried to use a transmitter and receiver with less complexity and less features with the only purpose of transmitting and receiving, we found FS1000A[3] transmitter and receiver, can be seen in Figure 4, but we decided to not use it because it utilizes a channel with the frequency of 433 MHz, where amateur usage was prohibited by Saudi Arabia regulation[4], as can be seen in Figure 5.

- Computer device with windows operating system, that will act as the device that needs authentication.

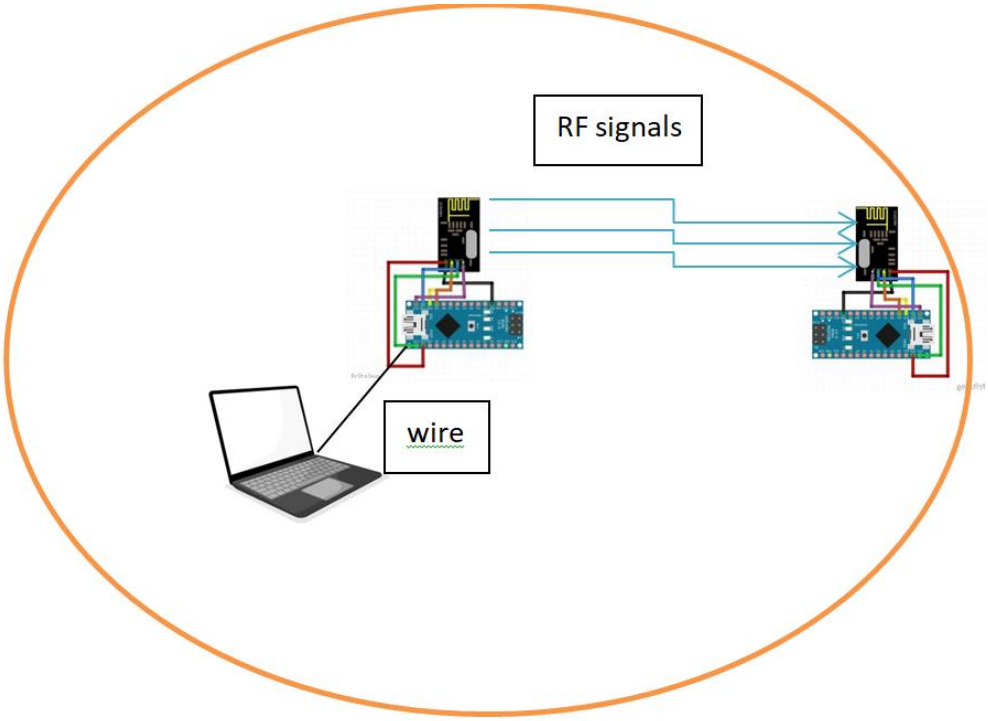


Figure 1 - Proximity-Based Authentication Design

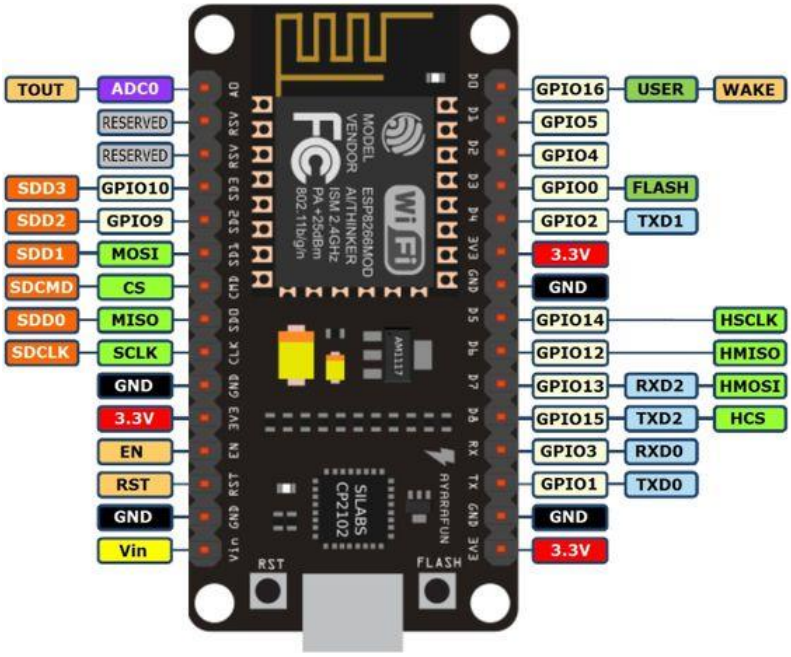


Figure 2 - NodeMCU 1.0

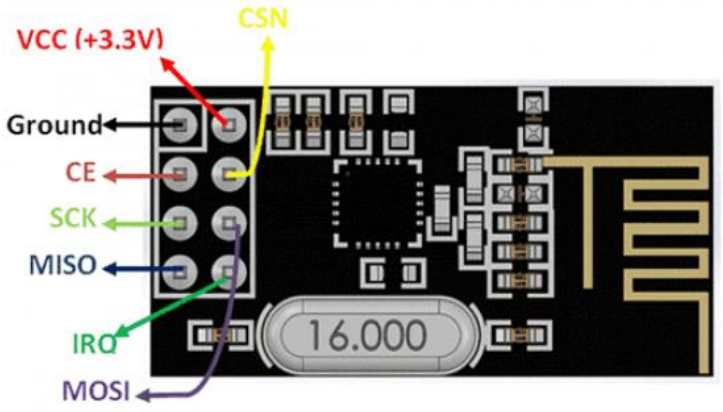


Figure 3 - NRF24L01

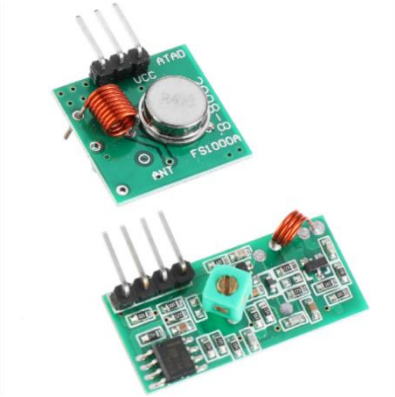


Figure 4 - FS1000A

432 - 434 MHz	Earth exploration-satellite (active)	GOV	5.279A 5.138 5.282 KSA07	Footnote Details...
	FIXED	GOV	5.276 5.138 5.282 KSA07	Footnote Details...
	MOBILE except aeronautical mobile	GOV	5.276 5.138 5.282 KSA07	Footnote Details...
	RADIOLOCATION	GOV	5.138 5.282 KSA07	Footnote Details...

Figure 5 - Usages of 432 to 434 MHz Bandwidth Taken from CST

APPROACHES

The physical implementation is very similar to the design mentioned earlier, as can be seen in Figure 6, the transmitter and receiver locations depend on the approach. This section will provide further information about the different approaches regarding coding and software for both issues mentioned in the *Problem Statement* section, with details regarding testing procedure, and the outcome of each one. All the next software is coded using the RF24[5][6] Arduino library.

Figure 6 - Physical Implementation

First Issue: Security

This issue is about the authenticity of the received signal, ensuring that the device is receiving signals that are produced by the user. To solve this issue, two abilities of the NRF24L01 transceiver have been utilized:

- Channel Assignment
  After initiating the transceiver, it provides the ability to choose from 125 different channels. Both transmitter and receiver must be on the same channel.
- Address
  The transmitter must sent to the receiver using a specified address, the address must be the same in both components.

The code for this part is fundamental in all the approaches detailed next, in both receiver and transmitter, which can be seen in the Appendix sections for the different approaches.

Before describing the approaches to solve the second issue, it is necessary to detail the testing procedure used for validating their usefulness to be utilized to find the approximate distance.

Testing procedure:

Only one testing method is used to validate the practicality of the collected values. Depending on the approach, one board will transmit and the other will receive the signals, and the responsibility of the person that conducts the experiments is deciding if the gathered values, like power received, can be utilized to find distance decision or they are not useful. For each approach, testing was conducted from different distances from 0.5 to 4 meters, with steps of 0.5 meters.

Second Issue: Proximity

This issue has been difficult to solve, different approaches have been tested. The details can be seen next:

- **Using the power received**
  The used transceiver provides limited received power information, in this case called Received Signal Strength Indicator (RSSI). There is only one bit given that can be utilized to as a hint about the received power, which can be accessed using `testRPD()`[7] method in the RF24 library. The purpose of this method, or bit, is to test whether a signal is greater than or equal to -64 dBm is present on the channel. The problem here is that we are sending and receiving with much higher power, using the `setPALevel`[8]`(RF24_PA_LOW`[9]`)` method that

is used to set the power amplifier level with an argument that corresponds to -6 dBm. Thus, the decision was that the values are fixed and with no usability.

- **Using the number of received messages as an indicator**
  This other approach seemed promising; however, it was not. The result was not consistent and cannot be used to take decisions. The architecture for this approach can be seen in Figure 7.

Figure 7 - Second Approach

- **Using the Round Trip Time (RTT) in millisecond**
  This method has been tested, where the transmitter is connected to the device and the receiver is a substitution of the user as can be seen in Figure 8. In this method, the transmitter sends a message and waits for a response, then it calculates the time from sending to receiving an acknowledgment, auto acknowledgement is enabled by default. The collected values have been very similar, as will be seen in the next section, which was between two and three milli seconds.

Figure 8 – Third and Fourth Approach

- **Using RRT in microseconds**

  A similar approach as the third and the same implementation as illustrated in Figure 8. The difference is that here the collected values are in micros, and these can be used to take an approximate decision. The values gathered from all distances are in the range of 2000 to 3000 microseconds, with large jumps. In order to finalize a decision and clear the random jumps (outliers), the average of the values have been used. 25 messages have been sent and their RRTs are used to find the average RRT, 100 sample averages from different distances are illustrated in Figure 9. The averages below 0.5 meters are all below 2150 microseconds of average RRT, so can be taken as threshold. To take a decision, to open the device for the user or not, the threshold (2150 microsecond) has been used to be the deciding anchor.

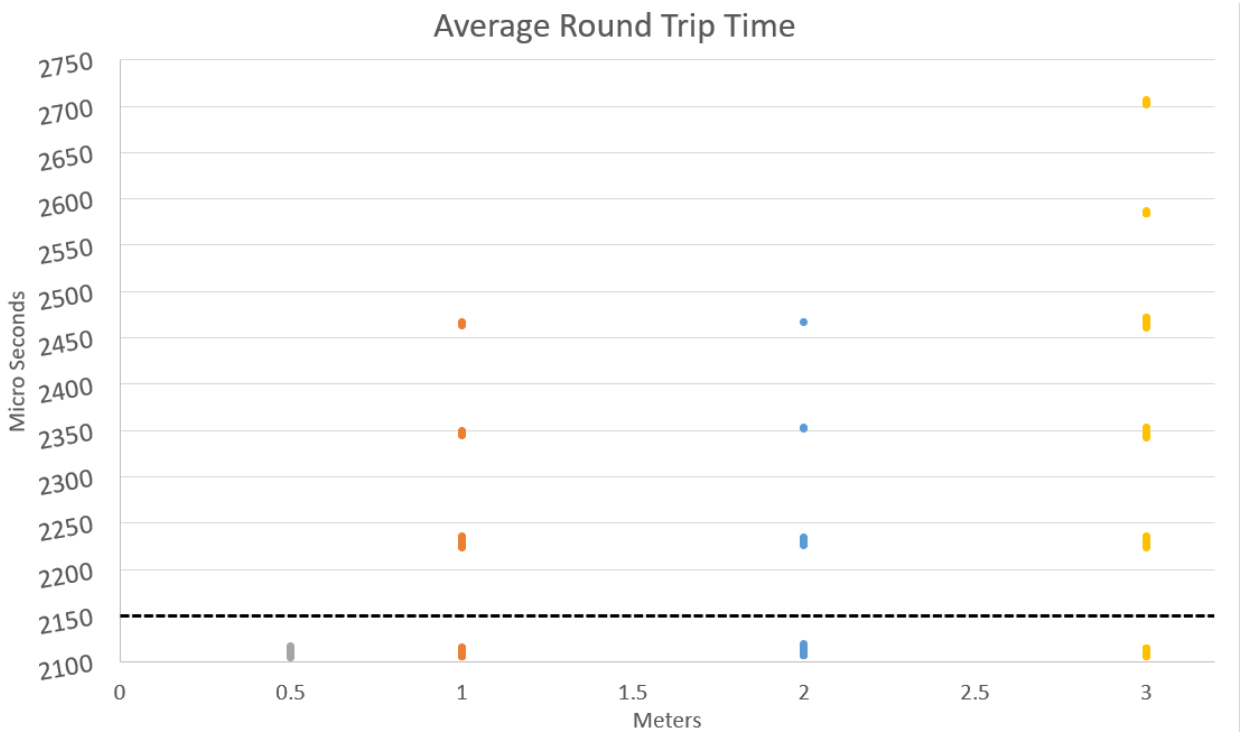


Figure 9 - Average RRT

The deciding procedure is as follows: If more than seven out of ten RRT averages is less than 2150 microsecond, that means the user is at 0.5 meter or less from the device, therefore, open the device. Another testing has been conducted to validate the accuracy of the decision procedure which is illustrated in Figure 10. The procedure from start sending until taking one decision takes from three to four seconds.

The code for this aapproach is in Appendix A and B. Appendix C provides the data plotted in Figure 9 and 10 as an Excel file.

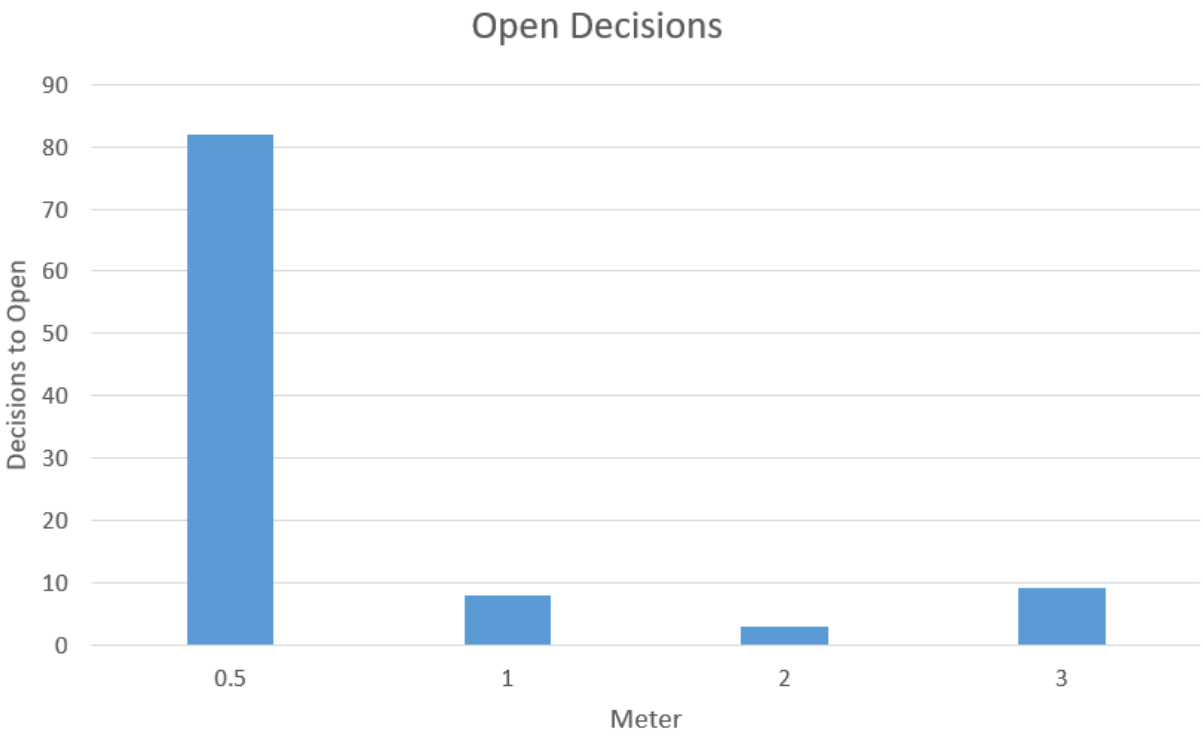


Figure 10 – Decisions to Open the Device

PROBLEMS FACED

- **Research**
  After researching the method to access NIC of an Android smart phone, with no approach given due to security concerns. Then moved to the idea of Software-defined radio[10], but it needed additional hardware. Finally, moved to execute our approach, which had taken a significant time. Also, the chosen transceiver, NRF24L01, is cheap and power efficient, but provides very limited power information. Other components with much more available power information might be used and could most likely correspond to much more accurate proximity decisions.
- **Integration**
  NRF24L01 connects to boards using at least seven pins to be able to operate successfully, IRQ is not essential. Four pins for SPI, power, ground, and a special pin called Ship Select Not (CSN). The number of pins and various codes provided to operate made the initialization of this component a confusing task. It took some time, but they worked at the end.[10]

- **Variance in The Testing Result**
  This was a critical issue. The testing has been done in three different rooms, two resulted in the outcome that have been discussed in fourth approach and one collected pure random values. After testing further by using a smart phone that transmits in the 2.4 GHz spectrum, the parameter affecting was concluded. Each room had different types of signals in different channels, which was affecting the collected results.

LESSONS LEARNED

- **NRF24L01 Transceiver**
  This transceiver is optimized merely for sending and receiving strings of 32 bytes as a maximum size. It is a very excellent option for various Internet of Thing (IoT) applications in the 100 meters range, depending on the atmosphere and transmitting power, like wireless sensors networks and embedded system communication networks. Also, support natively a short switching time between channels enabling frequency hopping technique for enhanced security and robustness.[2]
- **Scientific Testing Procedure**
  The first and second testing have been conducted in different rooms, but no data has been collected. The issue here is that it took time and place to perfume experiments and gathered data of each is much faster than doing the experiments again. Also, gathered data could display different relations in different situations. In the case of this project, unfortunately, data has been collected after deciding on a specific method. Also, significant segments of testing wireless systems, that variations of data can be inferred to other parameters and display much more randomness than wired systems.
- **Deciding to Utilize Merely a Parameter**
  In wireless systems, it is very difficult to ensure that a system is working properly using only one parameter. In the case of this project, the RRT varies extensively and to take an accurate decision using only this parameter will increase the processing power needed which in turn increases the processing time. Other parameters could be added easily and would make the decision procedure with higher accuracy, less computation, and processing time.

ADDITION TO IMPROVE OUTCOME

This feature can be integrated to regular devices, if NICs can be accessed, in any smart phone or PC. Messages as passwords can be encrypted and sent then received to be analyzed by the receiver to ensure validation of the message and the proximity of the transmitter. The power consumed by this proposed approach is negligible compared to current battery capacity, and more enhancement can be done as described next.

- **Power Control**

NRF24L01 support four different modes:[6]

- Power Up and Down Mode
  Using `powerDown()`[11] and `powerUp()`[12] methods. One case to integrate this: if a open decision has been decided, sleep for a period of time.
- Standby Mode[13]
- Transmit Mode
- Receiver Mode

- **Enhanced Security**
  Encryption can be easily added, making the password as the string message and each time encrypted using a different key. Also, as described earlier, frequency hopping is supported and can be used to be as an additional security procedure.
- **Additional Parameters**
  Other components can be added to provide more information. Components that exist in any ordinary smart digital device, like Global Positioning System (GPS), can be utilized to provide more accurate decisions with less computation time.

WORK DISTRIBUTION

Name	Responsibilities	Percentage
Hassan AL Nasser	Leader, parts provider, integration, testing, coding, documentation	55%
Yusef Seto	Testing	15%
Abdulmohsen Alanazi	Documentation and Testing	30%

CONCLUSION

In conclusion, this report details the Mobile Computing's course project (Proximity-Based Authentication). The goal is to authenticate a user to a his/her digital device using merely wireless signals. The problem has been broken down into two issue, security and proximity issues. Different approaches been tested to solve the proximity issue, using an Arduino board connected to NRF24L01 Transceiver. Promising results observed while testing the *fourth approach*, utilizing the RRT in microseconds. To overcome the jumps in the data, averaging has been used. Finaly, to take a decision, consistency of the averages has been considered. Finaly, a description of problems encountered, what has been learned, and what could be added to improve the results.

REFERENCES

[1] https://en.wikipedia.org/wiki/NodeMCU

[2] https://html.alldatasheet.com/html-pdf/1243924/ETC1/NRF24L01/112/1/NRF24L01.html

[3] https://html.alldatasheet.com/html-pdf/1568235/LPRS/FS1000A/111/1/FS1000A.html

[4] https://nfp.citc.gov.sa/

[5] https://www.arduino.cc/reference/en/libraries/rf24/

[6] https://nrf24.github.io/RF24/

[7] https://nrf24.github.io/RF24/classRF24.html#a821285f3b54553f4402eb3fd0ac6d6c1

[8] https://nrf24.github.io/RF24/classRF24.html#ab6a711a9cc14fb459a4a1698b8665d82

[9] https://nrf24.github.io/RF24/group__PALevel.html#gga1e4cd0bea93e6b43422855fb0120aacea7d8d09f4a047b7c22655e56c98ca010c

[10] https://forum.arduino.cc/t/simple-nrf24l01-2-4ghz-transceiver-demo/405123

[11] https://nrf24.github.io/RF24/classRF24.html#aa0a51923a09ba4f3478aba9be0f8a6a1

[12] https://nrf24.github.io/RF24/classRF24.html#a5cdaf47aa0edd6dca1b9a8bb7972a1a3

[13] https://nrf24.github.io/RF24/classRF24.html#a12cc453453c94969d4d3f0edb3778c83

APPENDIX A: TRANSMITTER CODE

```
#include <RF24.h>

#define CE_PIN   D4
#define CSN_PIN  D3

const byte ADDRESS[5] = {'R','x','A','A','A'};

RF24 radio(CE_PIN, CSN_PIN); // Create a Radio

double timeOfSending;
double timeACKReceived;
double sum;
double sumOfAverages;
double average;
double averageOfAverages;
bool rslt;
bool failed;
int counter;
int list[20];
char dataToSend[32] = "HelloHelloHelloHelloHelloHello";

int correct;
int iteration;

double start;
double end;
```

```
void setup() {

    Serial.begin(9600);

    while (!Serial) { }

    Serial.println("SimpleRx Starting");
    Serial.println("SimpleRx Starting");
    Serial.println("SimpleRx Starting");

    radio.begin();

    radio.setDataRate(RF24_250KBPS);
    radio.setPALevel(RF24_PA_LOW);
    radio.setChannel(0); // 2400 MHz
    radio.setAddressWidth(5);
    // this will print all the setting of transceiver
    radio.printDetails();
    radio.openWritingPipe(ADDRESS);

    correct = 0;
    iteration = 0;
}

int openCounter;
int notOpenConter;

void loop() {
```

```
failed = false;
openCounter = 0;
notOpenConter = 0;
for (int i = 0; i < 10; i++) {
  counter = 0;
  sum = 0;
  for (int j = 0; j < 25; j++) {
    timeOfSending = micros();
    rslt = radio.write( &dataToSend, sizeof(dataToSend) );
    if (rslt) {
        timeACKReceived = micros();
        sum = sum + (timeACKReceived - timeOfSending);
        counter++;
    }
    else {
        Serial.println("  Tx failed");
        failed = true;
    }
    delay(10);
  }

  average = sum / counter;
  if (failed) {
    Serial.println("failed transmission");
  } else {
  if (average < 2150)
    openCounter++;
  else
```

```
        notOpenConter++;
      }
    }

    iteration++;
    if (openCounter > 7) {
      Serial.println("OK");
      correct++;
    }
    else
      Serial.println("NOT");
    Serial.println(iteration);
    Serial.println("###############");
}
```

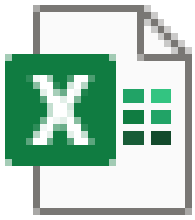
APPENDIX B: RECEIVER CODE

```
#include <RF24.h>

#define CE_PIN   D4
#define CSN_PIN  D3

const byte address[5] = {'R','x','A','A','A'};

RF24 radio(CE_PIN, CSN_PIN);

char dataReceived[32]; // this must match dataToSend in the TX
int counter = 0;
long int current;
long int delay0;

//===========

void setup() {
    Serial.begin(9600);

    while (!Serial) { }

    Serial.println("SimpleRx Starting");
    Serial.println("SimpleRx Starting");
    Serial.println("SimpleRx Starting");

    radio.begin();
    radio.setDataRate(RF24_250KBPS);
```

```
    radio.setPALevel(RF24_PA_MIN, 0);
    radio.setChannel(0); // 2400 MHz
    radio.setAddressWidth(5);
    radio.printDetails();
    radio.openReadingPipe(1, address);
    radio.startListening();
}
//=============

void loop() {
    if ( radio.available() ) {
        radio.read( &dataReceived, sizeof(dataReceived) );
        counter++;
    }
    // printed to ensure that the receiver is fucntioning
    Serial.println(counter);
}
```

APPENDIX C: COLLECTED DATA

The collected data for the Figures 9 and 10 can be seen in this Excel file, double right clicks on the file will open the fire:



data.xlsx