THOMPSON RIVERS UNIVERSITY

COMP 4910

Distance Measurement using Radio Frequency Signal

Instructor: Kevin O'Neill

Contact: Mahnhoon Lee

Company: Pro-Active Safety Systems Technologies (PASST)

Hedao Tian (T00531517)

Lei Zhu (T00531522)

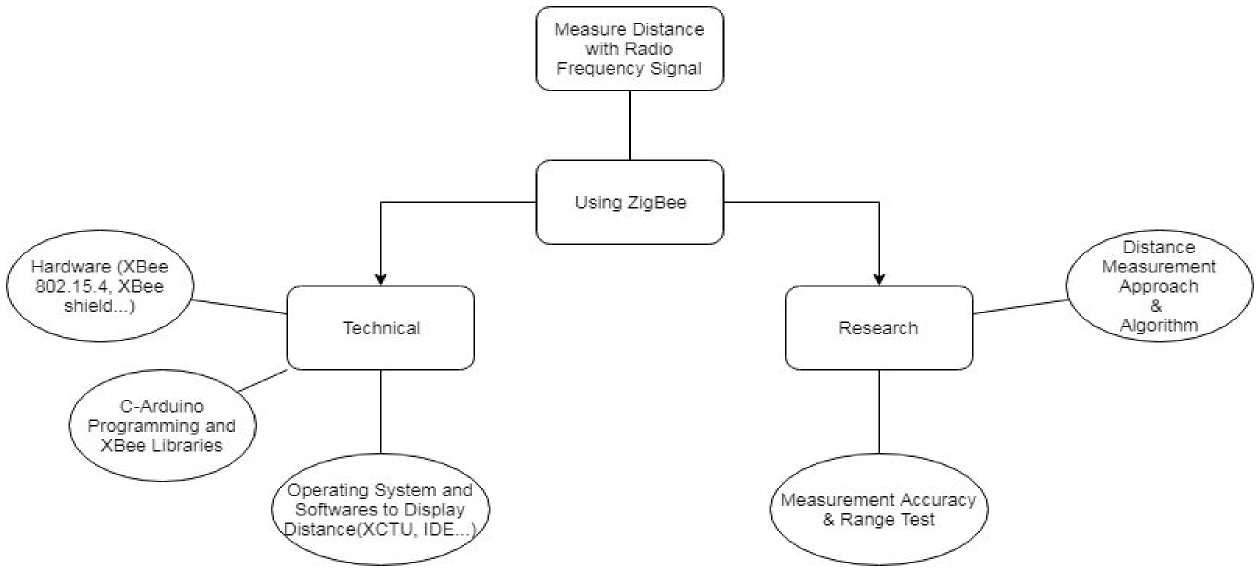
ZhenYu Wang (T00059541)

Table of Content

Requirement

Range information can be used in many applications such as collision detection for automobiles. There are lots of ways to help us measure the distances between two objects such as GPS, sound, camera and ZigBee.

Our project focus on quickly measures the distance between two stationary objects in certain environment using the ZigBee radio frequency module. The maximum range measurement by using ZigBee should be determined. We also need to provide an accurate algorithm in the program with its accuracy. After doing research on the distance measurement field with radio frequency, we separated the requirements into technical and research parts:


Measure Distance with Radio Frequency Signal
Using ZigBee
Technical
Research
Hardware (XBee 802.15.4, XBee shield...)
C-Arduino Programming and XBee Libraries
Operating System and Softwares to Display Distance(XCTU, IDE...)
Distance Measurement Approach & Algorithm
Measurement Accuracy & Range Test

Technical

To achieve the distance measurement, we should be able to send and receive signal and to measure the signal strength to from arduino board, meanwhile the distance should be calculated and displayed on arduino IDE (Integrated development environment) which is the programming environment of arduino. Thus below hardwares are required:

● XBee 802.15.4 (Series 1/Series 2) x 2	//wireless communication standard
● XBee shield x 2	//Bridge between XBee and Arduino
● Arduino Uno R3 x 2	//Microcontroller board for programming
● Arduino USB x 2	
● Arduino 9V Battery (cable included) x 1	//Remote XBee power supply (9-12 Volts)
● Laptop (Mac OS X/Windows 8 or later) x 1	//Platform to run IDE and display distance

Certain softwares are also required:

● XCTU	//For XBee channel, ID, API....configuration
● Arduino IDE (with XBee libraries)	//Programming environment for Arduino
● Python	//Terminal for end users

Research

Once the RSSI which is the measurement of signal strength can be captured, our group should then provide the algorithm to calculate the RSSI value into distance with meters since the transmission limitation of ZigBee is 10 to 100 meters (https://en.wikipedia.org/wiki/Zigbee). Thus our group needs to measure the distance between the range to gain enough samples to generate the pattern. We will be reading certain papers and documentations to understand what has been done and which approaches are failed. To measure the distance, we need certain things:

- Ruler (Length of 20m)
- Anaconda Jupyter/ Weka //To visualize the data

*Different power output of ZigBee and environmental characteristics will affect the signal transmission range.

Details of the technologies we require to use

XBee 802.15.4:

Wireless technology which is also called ZigBee or XBee Series 1, is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power, wireless sensors network . It transfer frames by sending 8 bit messages (V. K. Sehgal et al., 2009).

XBee shield:

The shield form-factor mates directly with any dev board that has an Arduino standard footprint and equips it with wireless communication capabilities using the popular XBee module. As D'Ausilio and A. Behav Res said, Xbee shield allows multiple Arduino boards to communicate wirelessly.

Arduino UNO Rev3:

As a microcontroller, this board can be enrolled into our system as the central controllers which control the wireless on real time after the code is uploaded into it (Iman, Sami H and M. S., 2015).

Arduino IDE (Integrated development environment):

The Arduino IDE provides a simplified integrated platform which can run on regular personal computers and allows users to write programs for Arduino using C or C++ (Verma, 2017). Our team needs to be familiar with the libraries to have a better understanding about the system.

Optional:

Weka:

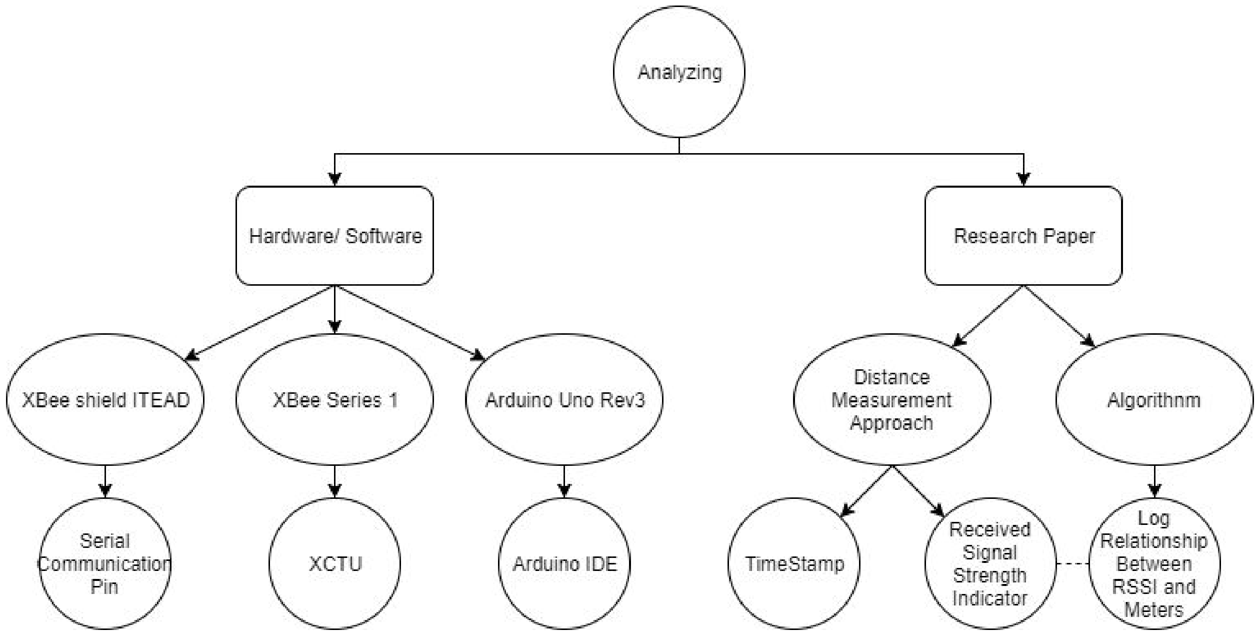
Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes (https://www.cs.waikato.ac.nz/ml/weka/). We might use Weka to visualize the data and try to generate the function.

Anaconda Jupyter:

By using Jupyter, we can visualize the data more customize rather than using Weka.

Analysis & Testing

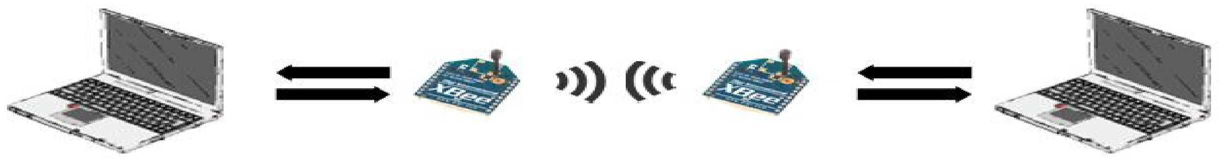
Since this is a research based project, our group spent lots of time on doing research and learning new hardware, software and algorithms. First we need to build the hardwares with software needed, then study and use some existing research paper knowledge. We divided the analyzing into two parts as shown below:



Hardware/ Software

XBee Series 1 (802.15.4)

Digi's XBee 802.15.4 Wireless Modules (formerly called XBee Series 1). IEEE 802.15.4 networking protocol for fast point-to-multipoint or peer-to-peer networking. They are designed for high-throughput applications requiring low latency and predictable communication timing. As Florian et al. describes, IEEE 802.15.4 is a popular wireless communication standard developed for simple low-power and low-cost wireless applications. One of these applications is local positioning, which consists in inferring the position of a device in real time (2014).



XBee Series 2

The Series 2 uses a microchip from Ember Networks that enables several different flavors of standards-based ZigBee mesh networking (Faludi, 2010).

Series 1 vs Series 2

	XBee Series 1	**XBee Series 2**
Indoor/Urban range	up to 100 ft. (30m)	up to 133 ft. (40m)
Outdoor RF line-of-sight range	up to 300 ft. (100m)	up to 400 ft. (120m)
Transmit Power Output	1 mW (0dbm)	2 mW (+3dbm)
RF Data Rate	250 Kbps	250 Kbps
Receiver Sensitivity	-92dbm (1% PER)	-98dbm (1% PER)
Supply Voltage	2.8 – 3.4 V	2.8 – 3.6 V
Frequency	ISM 2.4 GHz	ISM 2.4 GHz
Operating Temperature	-40 to 85 C	-40 to 85 C
Filtration Options	PAN ID, Channel & Source/Destination	PAN ID, Channel & Source/Destination

To conclusion, there are three types of ZigBee offer in market, such XBee Series 1 that able to support up to 100 meters, XBee Series 2 that able to support until 125 meters and XBee Pro Series 1 able to support coverage up to 1.5 kilometers(Salleh et al, 2013). To have a better range on our peer-to-peer experiment, XBee Series 1 Pro is the best option.

*http://icircuit.net/xbee-series-1-vs-series-2/289

XBee Series 1 vs Series 1 Pro

The XBee and XBee-PRO RF Modules were both engineered to meet IEEE 802.15.4 standards and support the unique needs of low-cost, low-power wireless sensor networks. The Pro series have the same pinout and command set of the basic series with an increase output power of 60mW.

*https://www.digi.com/resources/documentation/digidocs/PDFs/90000982.pdf

	XBee Series 1	XBee Series 1 Pro
Range	300ft (100m)	1 mile (1500m)
Voltage Regulator	3.3V @ 50mA	3.3V @ 215mA
Transmit Power Output	1 mW (0dbm)	60 mW (+18dbm)
RF Data Rate	250kbps	250kbps

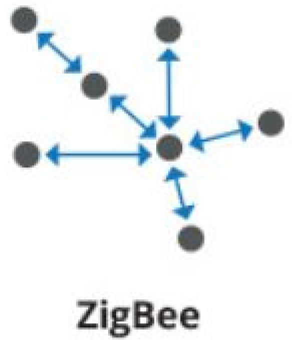
*https://www.sparkfun.com/products/8742

XCTU

XCTU is a free multi-platform application designed to enable developers to interact with Digi RF modules through a simple-to-use graphical interface.

There are certain different Radio Frequency modules such as XTend, XLR and XBee RF modules which are the module we are working on.

By using XCTU, we can configure the XBee's certain options to achieve a ZigBee communication module (Point to point or Broadcasting):



- Channel
  - Frequency band that your XBee communicates over.
- Personal Area Network ID (PANID)
  - The network ID is some hexadecimal value between 0 and 0xFFFF. XBees can only communicate with each other if they have the same network ID.
- MY address
  - Addresses for each XBee in your network. Each XBee in a network should be assigned a 16-bit address (again between 0 and 0xFFFF)
- Destination address (DH & DL)
  - The destination address defines which XBee your source XBee is talking to.

*https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu/configuring-networks

ARDUINO UNO REV3

Arduino Uno is a microcontroller board based on the *ATmega328P(*)*.

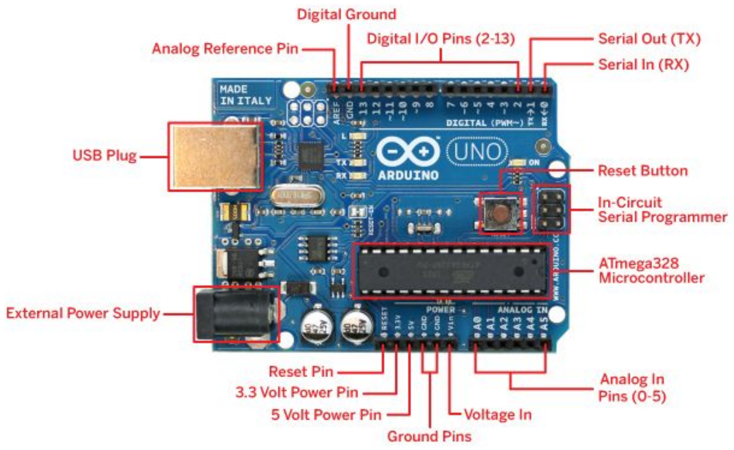
* The ATmega328 is a single-chip microcontroller created by Atmel in the megaAVR family.

Comparing with other microcontroller board are commonly used, Arduino UNO is:

- Simple computer that can run one program repetitively
- Easy to use

  Structure:

  It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.



Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

*https://www.arduino.cc/en/Guide/Environment

Serial

Used for communication between the Arduino board and a computer or other devices. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

SoftwareSerial Library

To build the connection between Arduino and XBee, Arduino and IDE, we use SoftwareSerial Library. The Arduino hardware has built-in support for serial communication on pins 0 and 1 (which also goes to the computer via the USB connection).

The library has the following known limitations:

- If using multiple software serial ports, only one can receive data at a time (USB connection exclusive).
- Maximum *RX* speed is 57600 bps (Bytes Per Second)

*TX and RX are abbreviations for Transmit and Receive, respectively. Units are in Bytes.

//How to set up software serial:

```
SoftwareSerial serial2(8, 9); // RX, TX
```

//How to start serial:

```
Serial.begin(9600);  //Set serial data transmission rate to 9600
serial2.begin(9600); //Set software serial data transmission rate to 9600
```

Xbee-arduino Library

Arduino library for communicating with XBees in API mode, which includes support for the majority of packet types, including: TX/RX, AT Command, Remote AT, I/O Samples and Modem Status.

//How to set XBee on Serial

```
XBee xbee=XBee();
xbee.setSerial(serial2);
```

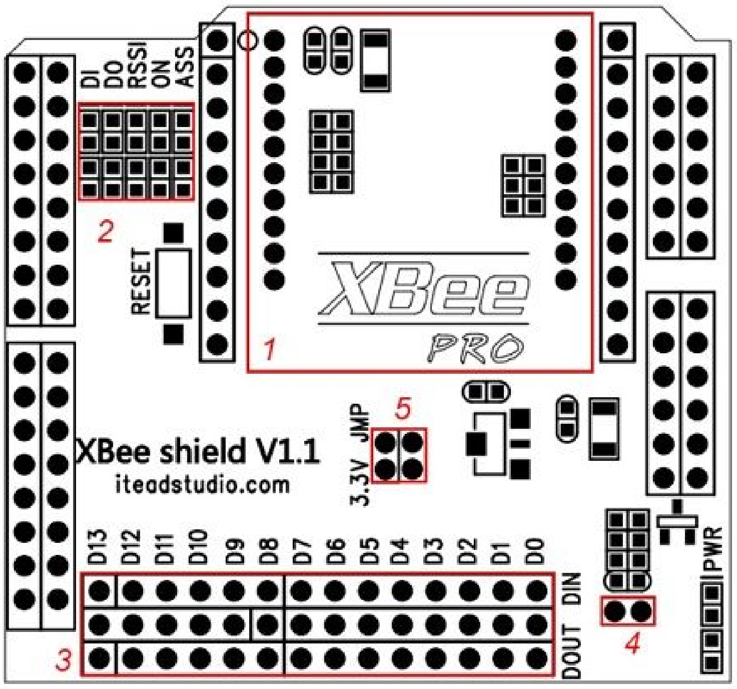
```
//How to send a frame
XBee xbee = XBee(); //Create xbee object by using the xbee library
uint8_t payload[] = { 'H', 'i' }; //Create a payload to send message
Tx16Request tx16 = Tx16Request(0xFFFF, payload, sizeof(payload)); //Create a frame to transmit
xbee.send( tx16 );  //Transmit the frame
//How to receive
xbee.readPacket(100); //Receive packet every 0.1 second
rssi = rx16.getRssi(); //Get the RSSI of the packet
```

*https://www.arduino.cc/reference/en/language/functions/communication/serial/

*https://www.arduino.cc/en/Reference/SoftwareSerial

XBee shield ITEAD

The XBee Shield simplifies the task of interfacing an XBee with your Arduino. This board mates directly with an Arduino Pro or USB board, and equips it with wireless communication capabilities using the popular XBee module.



Zone	Description
1	XBee Socket
2	Indication LED
3	Serial communication pin select
4	Wireless program Arduino jumper
5	3.3V operation voltage jumper(When operated in 3.3V,install the jumper)

Serial Communication Pin

In zone 3, two jumpers are set to connect XBee_DIN, XBee_DOUT to Digital pin of Arduino. When XBee communicate to Arduino hardware serial ports, set the jumper connect DIN to D1, DOUT to D0. For example, if we set SoftwareSerial serial 2 (8, 9);, this means we need to set Dout and Din to D8 and D9 correspondingly.

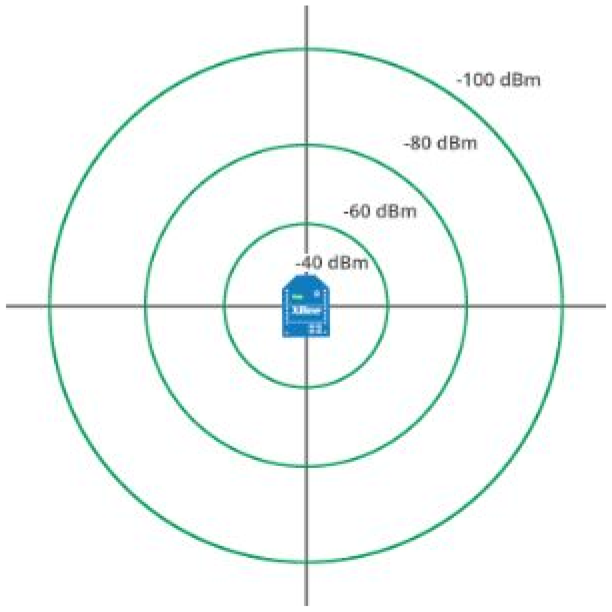
Research Paper

Distance Measurement Approach

The most important characteristics for distance measurement by using wireless communication are Received Signal Strength Indicator (RSSI), Time of Arrival (ToA), and Time Difference of Arrival (TdoA) (Mare, Cvetan and Vladimir, 2008).

As Jianwu and Lu said, distance measurement based on RSSI, featuring low communication overhead and low complexity, is widely applied in the range-based localization of the Wireless Sensor Networks (2009). And some researcher online post their research result on distance measurement with ToA which shows a poor result since the time doesn't change much while the distance is changing. Thus we decided to move to measuring distance by searching for the changing pattern of RSSI.

*Received Signal Strength Indicator (RSSI) measures the amount of power present in a radio signal. The RSSI is measured in dBm. A greater negative value (in dBm) indicates a weaker signal. Therefore, -50 dBm is better than -60 dBm.



https://www.digi.com/resources/documentation/Digidocs/90001456-13/concepts/c_rssi_pin_and_signal_strength.htm

There are also some other distance measurement approaches, such as Time of Arrival which calculates the time spent during the signal transmission. However, this approach doesn't work effectively. Thus we studied more into RSSI distance measurement.

Approaches Comparison

There are also two other options for precise distance measurement using non-arduino functions, such as Time of Arrival and Time Difference of Arrival.

The TDOA has a testing range of 300 meters with 0.09 meters accuracy as S. Schwarzer says(2008). And the TOA simulation results show that the proposed TOA estimation algorithm provides an accuracy of 0.5 m at a signal-to-noise ratio (SNR) of 8 dB and achieves an SNR gain of 5 dB as compared with the existing algorithm(Cheon, Hwang, Kim and Jung, 2016). The maximum testing range in TOA is 20 meters. For researchers using RSSI approaches, they draw the conclusion that the measurement error of Gauss model is 2 meters within 20 meters without consideration of circumstances effects (Jianwu, 2009). Li, Ya and Guoshao's experiment shows a 0.4 meters error when testing in 5 meters.

According to our research schedule and the limitation of workload, RSSI is the most valuable option for our team to investigate deeply, and both TDOA and TOA requires large amount of high-level mathematical knowledge.

Name	Range	Accuracy	Note
RSSI (Received Signal Strength Indicator)	20 meters 5 meters	2 meters 0.4 meters	Accuracy can be affected by testing environment but also can be improved by using least squares algorithm
TDOA (Time-difference-of-arrival)	300 meters	0.09 meter	Different Hardwares (Chips with antenna). Multiple XBee chips with 16 channels
TOA	20 meters	0.5 meter (18.25 meters before optimization)	Different hardware (Field-Programmable Gate Array with ADC/DAC and antenna).

RSSI Algorithm

As Li, Ya and Guoshao say, the algorithm below shows a log relationship between RSSI and distance (2014):

$$\mathbf{RSSI = A - 10n\ lg\ (D)}$$

In the Algorithm, RSSI can be measured in real time when every frame is received by receiver. D means the distance between transmitter and receiver, and A is the RSSI value when two XBees are 1 meter away from each other. A can change when the testing environment is changed.

Thus there is only one unknown value "n" called path loss value, which can be calculated as below:

$$n_i = \frac{A - \mathrm{RSSI}_i}{10 \lg d}$$

Path loss value will be measured certain times in specified environment to get its average value in order to generate our own algorithm.

$$\bar{n} = \frac{1}{N} \sum_{i=1}^{N} n_i$$

*Different algorithm modules should be built for different environment since the environment effectiveness might be different (Li, Ya and Guoshao, 2014).

To train the algorithm and generate below calculation, we will get average n value for accurate distance transformation from RSSI(d0=1 meter):

$$d = d_0 \times 10^{\left(\frac{\bar{A} - \mathrm{RSSI}_i}{10\bar{n}}\right)}$$

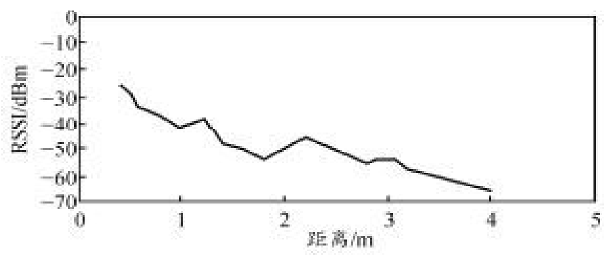
Thus a large amount of data collection is necessary in this distance measurement approach. In this case, the researcher did data collection in different environment with different versions of devices for better distance measurement result.

First Data Collection (In Door)

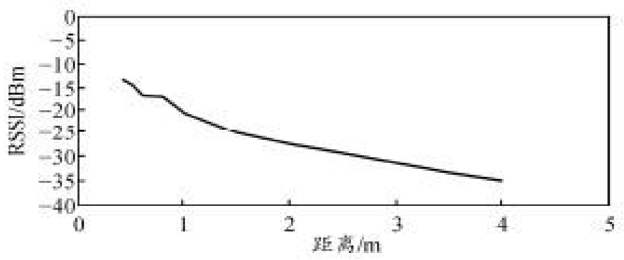
Our first data collection happened in Old Main building in front of Starbucks. We measured the distance by using rulers and did data collection at each spot from 1 meters to 18 meters, holding both XBee 1.2 meters away from the ground as Li, Ya and Guoshao said. The signal strength became stronger comparing with putting the devices on the ground, because the signal reflected from ground can affect the signal transmission. To have accurate measurement result, RSSI should be stable as possible (2014).

In all our data collection events, all personal electronics are in flying mode, and events were suspend when interference is around us(cars, people, airplane...).Below are two graphs they drew according to the 60 data collected when the distance of devices are 0 or 1.2 meters away from the ground.

0 meters away from the ground:



1.2 meters away from the ground:

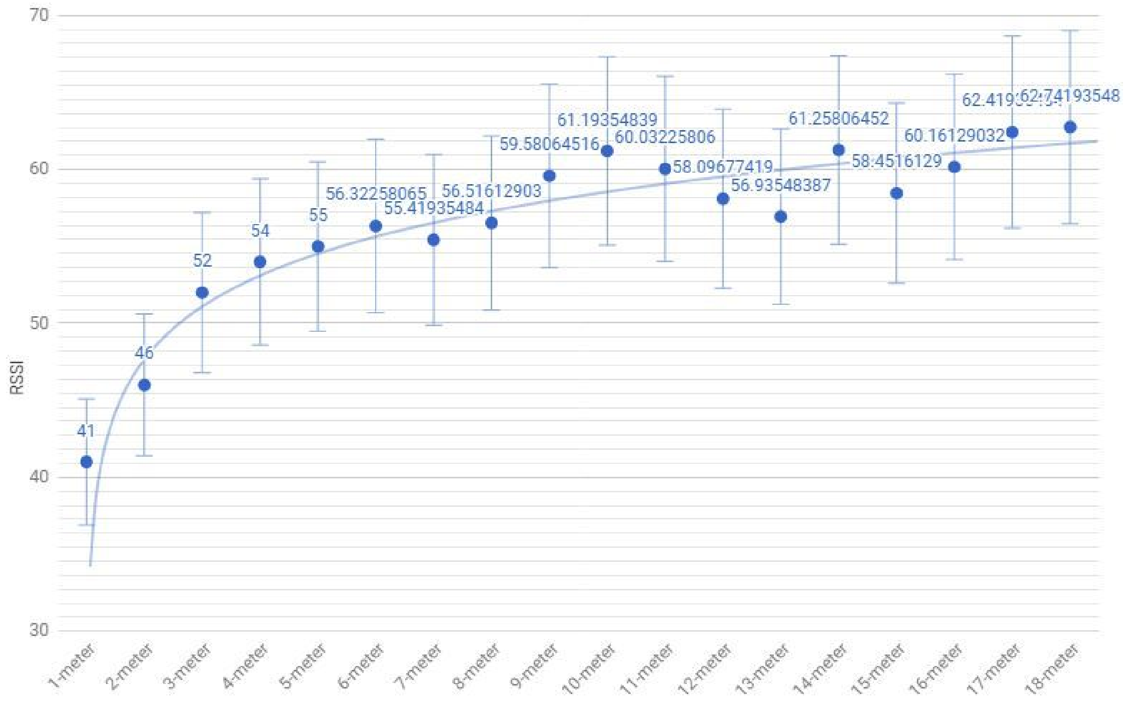


By monitoring the data collection process, we found the RSSI became more unstable when distance is above 5 meters. On the other hand, the RSSI in 5 meters are stable in most of the time. We collected 30 RSSI values from each testing spot and set up the values in 5 meters to the values that show up the most frequently at each spot.

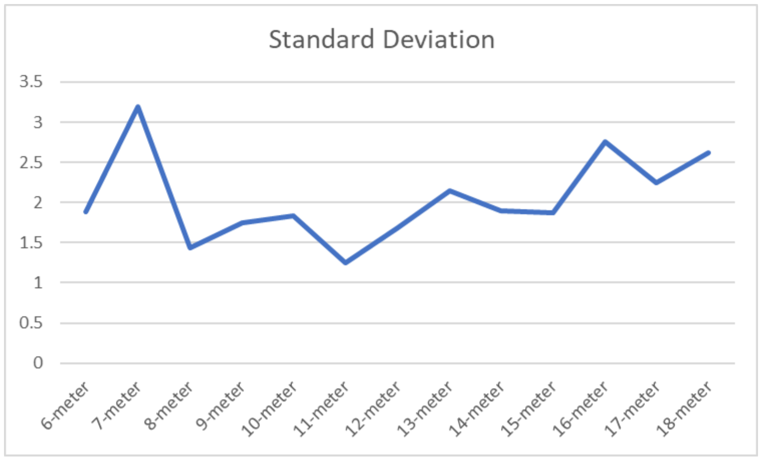
Testing Set Up:

Attributes	Setting
Frequency	1000 Million seconds/Frame
Payload Size	2 Bytes/ 16 Bits
Location	Old Main Starbucks
Environment	Quite, Indoor

Then we drew the graph below showing the average value of RSSIs measured from each spot:



We also measured standard deviations for those unstable values, the result is shown below:



Distance (meters)	6	7	8	9	10	11	12	13	14	15	16	17	18
Standard Deviation	1.88312342	3.19138625	1.43459559	1.74688663	1.83338220	1.25123594	1.68037374	2.14375813	1.89679979	1.87685392	2.75797962	2.24757934	2.62002216

By looking at the standard deviation, we can see the discrete ratio became higher when testing distance goes further. And the testing range is limited to 5 meters only. Thus we got feedback from our client and did evolution of the system.

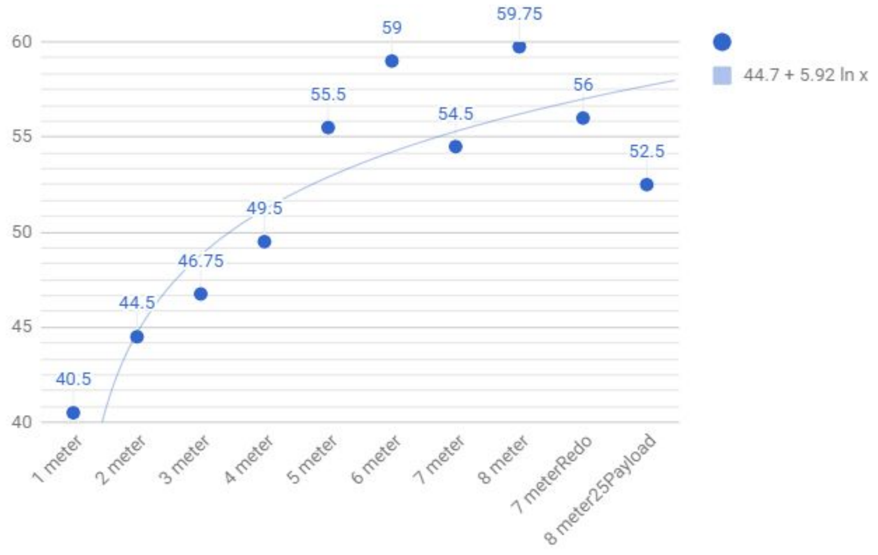
Evaluation

Since the signal result from first data collection was not stable. And the data is not recorded properly during the test. So we discussed the result with our client and did system evaluation. **See details in implementation part at page 36.**

Attributes for Evaluated System	Setting
Frequency	Adjustable
Payload Size	Adjustable
Location	Old Main Starbucks/ Parking Area
Environment	Quite, Indoor/ Cloud Free

Second Data Collection (In Door)

After implementing getting RSSI in a cycle instead of getting RSSI from only one side, we are able to get more accurate result in to .25, .5, .75 which means there are three more options between two integers. We did the same test on XBee Series 1 with frequency 500 million seconds with payload size of 1 byte. And we calculated the standard deviation on each distance, and the most unstable signal happened on 8 meters. Then the payload size was increased to 25 bytes to redo the 8-meter data collection, and the signal become much more stable (standard deviation was decreased from 2.95 to 1.12):



Distance (Meter)	1	2	3	4	5	6	7	8	8 (Payload 25)
Standard Deviation	1.35858 85	1.41448 8189	1.21533 8645	1.56251 5537	1.41021 295	1.24635 0605	0.87539 3374	2.94673 7639	1.121691 62

Third Data Collection (DayTime_Nocloud)

As Li, Ya and Guoshao said, different environment can have different radio frequency interference. To calculate distance based on RSSI, different formula needs to be trained to fit typical environment (2014).

To show a better pattern for RSSI, we moved the experiment area to an open area next to TRU new residence. To do deep research, we adapted both XBee S1 and XBee S1 Pro for further testing range based on different transmission power level.

Power levels of XBee S1:

Setting	Power level
0(Lowest)	-5 dBm
1(Low)	-1 dBm
2(Medium)	+1 dBm
3(High)	+3 dBm
4(Highest)	+5 dBm

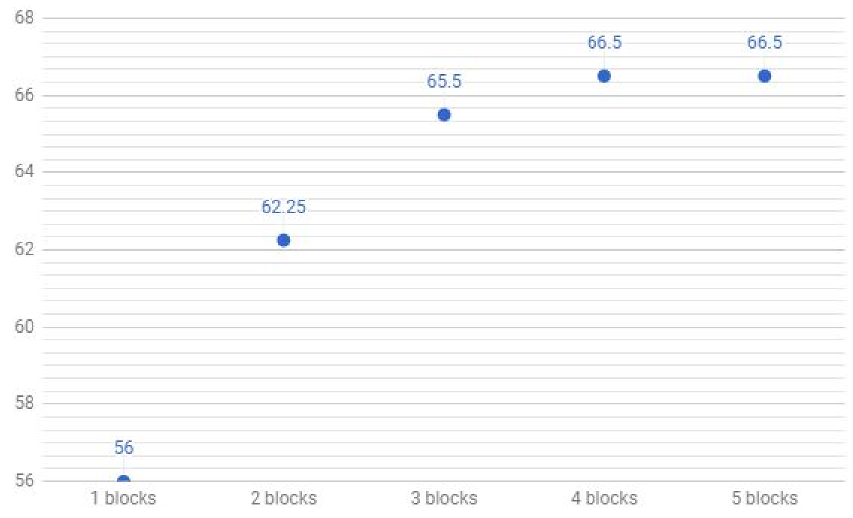
Power levels of XBee S1 Pro:

Setting	Power level
0(Lowest)	0 dBm (approximate)
1(Low)	+12 dBm (approximate)
2(Medium)	+14 dBm (approximate)
3(High)	+16 dBm (approximate)
4(Highest)	+18 dBM

RSSI testing started from measurement Lowest, Medium and Highest for both devices. And below testing happened due to RSSI performance. Before the testing, we modified payload size to see which value provides high performance for the whole system (efficiency, stability, range). The collection setting was RSSI measurement by blocks (2.75 meters each). Payload size for XBee S1 is 15 and XBee S1 Pro with 40. Power levels were configured in XCTU before the testing.

*Testing range was from 1 to 10 blocks for XBee S and 1 to 1, any distance was not mention in the graph means it's out of testing range in that setting.

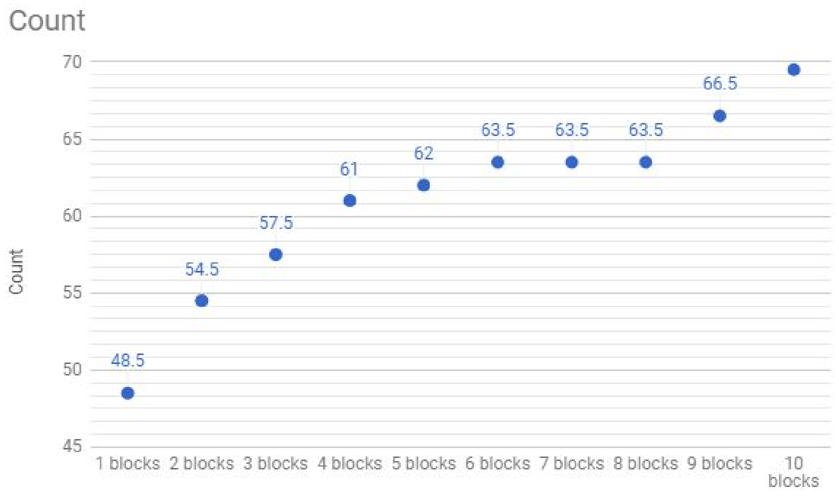
- Data collection for S1 under below power levels:
  - Lowest(Medium Value)



Note: Testing range is too short if power is low

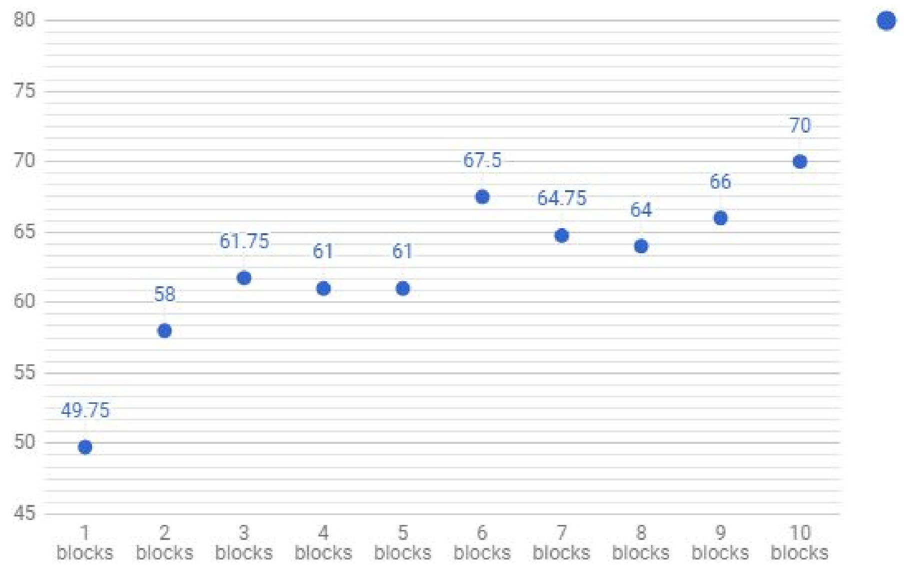
Distance	1 blocks	2 blocks	3 blocks	4 blocks	5 blocks
Standard Deviation	1.081299699	0.8601340061	0.8763609271	0.8135070899	0.8204225864

  - Medium(Minimum Value)



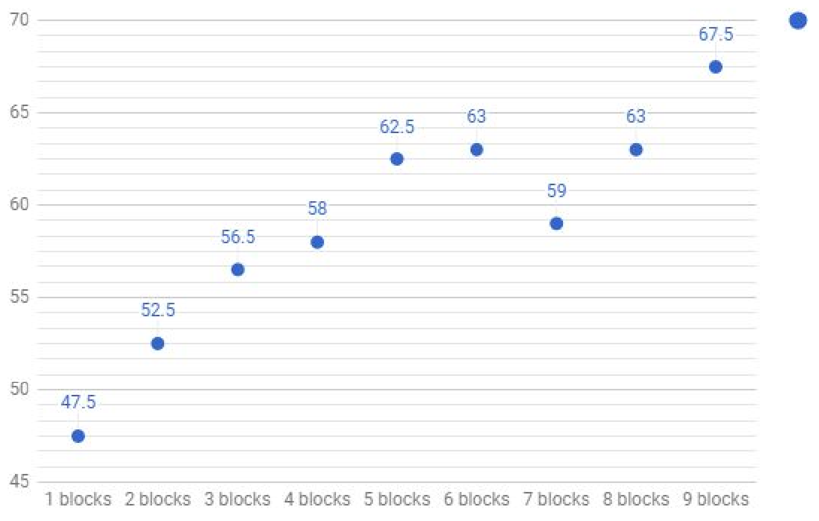
Distance	1 blocks	2 blocks	3 blocks	4 blocks	5 blocks	6 blocks	7 blocks	8 blocks	9 blocks	10 blocks
Standard Deviation	0.8049985086	0.4160447336	0.4347218086	0.6917594043	0.6955772305	0.8495595402	0.5561037785	1.473183071	0.7207576052	1.139331788

- High(Medium Value)



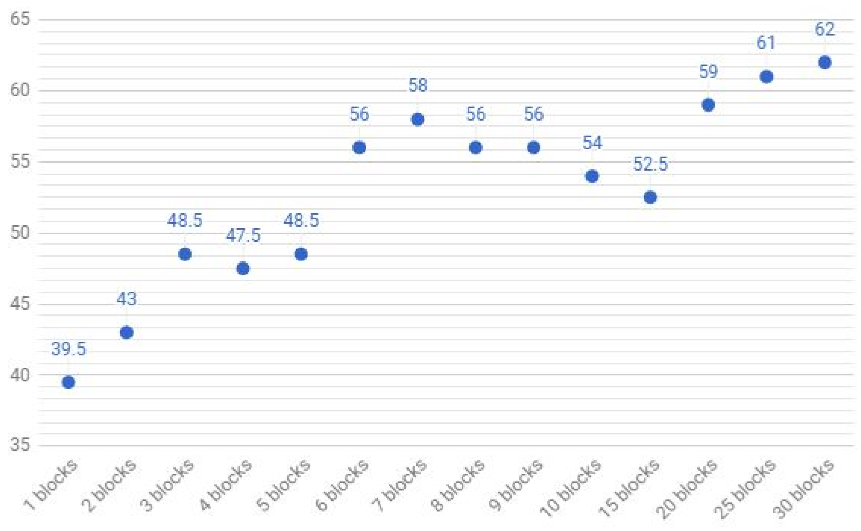
Distance	1 blocks	2 blocks	3 blocks	4 blocks	5 blocks	6 blocks	7 blocks	8 blocks	9 blocks	10 blocks
Standard Deviation	1.010502756	0.85552376	1.265887733	0.818224618	1.178790916	1.928378611	1.363647278	1.194945806	0.8255711941	1.173236145

- Highest(Median Value)



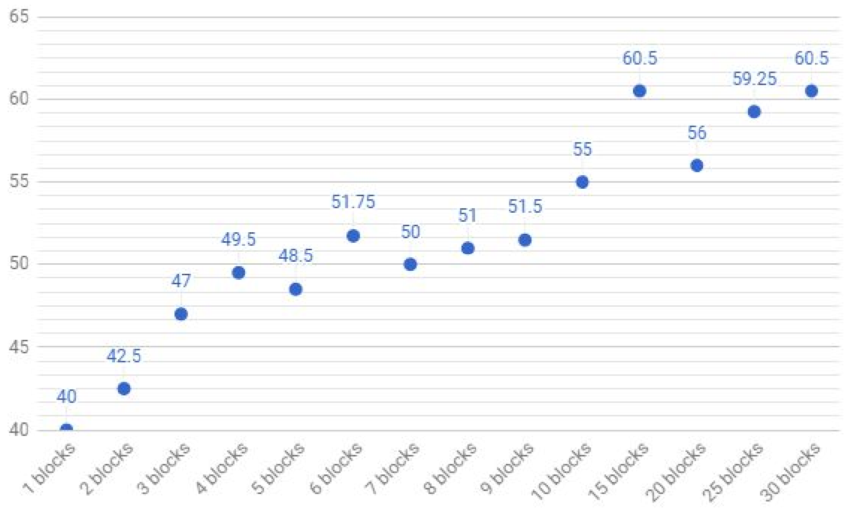
Distance	1 blocks	2 blocks	3 blocks	4 blocks	5 blocks	6 blocks	7 blocks	8 blocks	9 blocks
Standard Deviation	0.7461766956	0.9640428056	0.6825094937	0.5551505115	0.9893143777	2.005976241	0.3791903358	0.870742264	1.30080375

- Data collection for S1 Pro under below power levels:
  - Lowest(Medium Value)



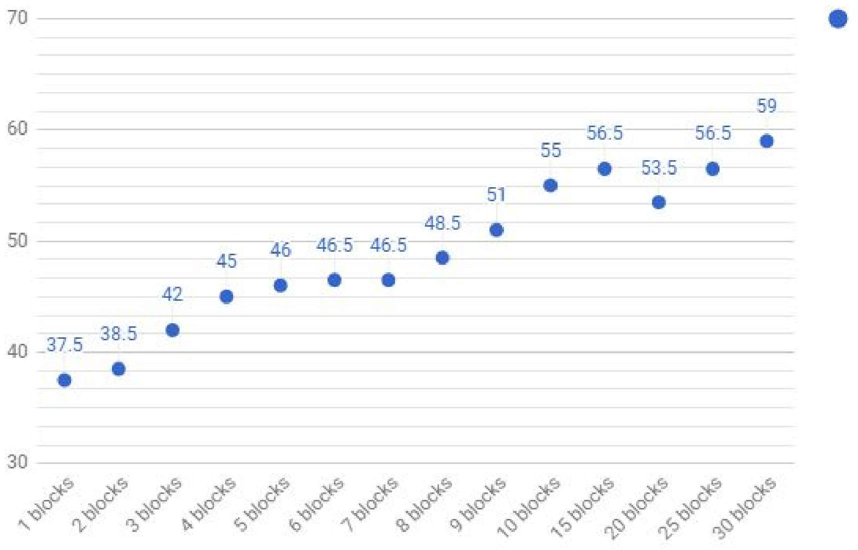
Distance	1 block	2 block	3 block	4 block	5 block	6 block	7 block	8 block	9 block	10 block	15 block	20 block	25 block	30 block
Standard Deviation	0.694 2052 319	1.283 8358 4	0.808 2816 397	0.335 6206 831	0.340 7369 879	1.376 1487 24	1.537 1804 48	0.905 8426 174	0.925 5857 818	0.867 8175 757	0.626 8475 517	0.904 2820 294	0.554 5777 648	1.151 6078 17

  - Low(Medium Value)



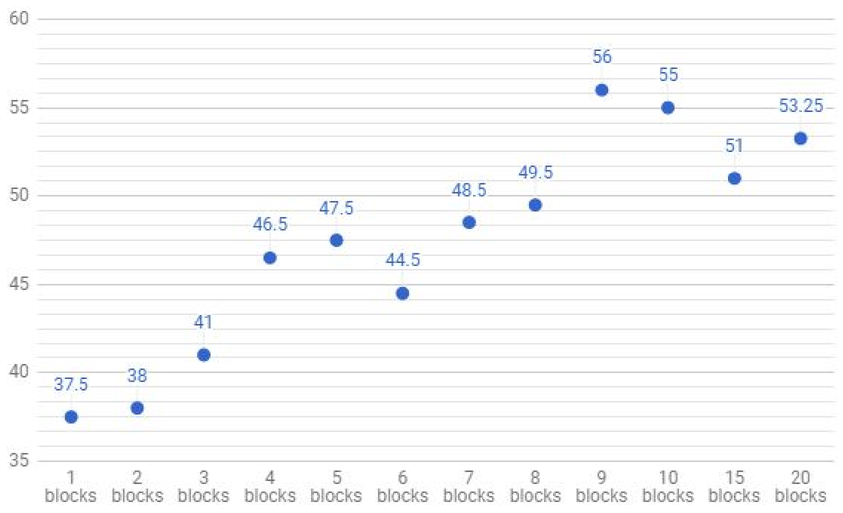
Distance (blocks)	1	2	3	4	5	6	7	8	9	10	15	20	25	30
Standard Deviation	0.646 3172 508	1.273 8958 88	0.970 7589 203	0.752 5380 783	0.550 8083 428	0.952 6575 97	0.348 9289 504	0.436 5860 106	0.683 2851 023	0.668 0776 03	1.795 6318 62	0.810 2451 327	1.255 3838 86	0.870 2554 986

- Medium(Minimum Value)



Distance	1 block	2 block	3 block	4 block	5 block	6 block	7 block	8 block	9 block	10 block	15 block	20 block	25 block	30 block
Standard Deviation	0.7611691869	0.6138698219	1.257154666	0.6939508604	0.4691017983	0.6580123297	0.5880159112	0.8799396487	1.120274129	0.8715124201	0.7954673856	1.048909846	0.7572623902	1.01301558

- Highest(Median Value)

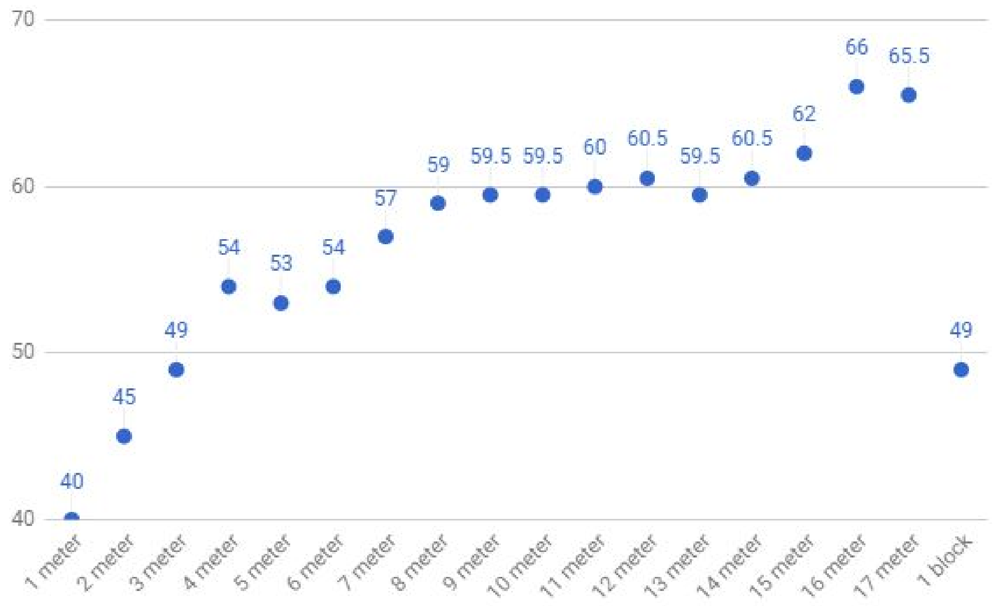


Distance	1 blocks	2 blocks	3 blocks	4 blocks	5 blocks	6 blocks	7 blocks	8 blocks	9 blocks	10 blocks	15 blocks	20 blocks
Standard Deviation	0.1936491673	0.4382811831	0.5702115773	1.404668367	1.687441148	2.141383338	0.7381359175	0.8511789763	1.700863855	1.276139128	1.212109335	1.430499103

Note: Signal became unstable if power level is strong

According to the graphs shown above, both devices were showing a good pattern when power levels are medium. And their minimum values in each sample set have the best pattern.

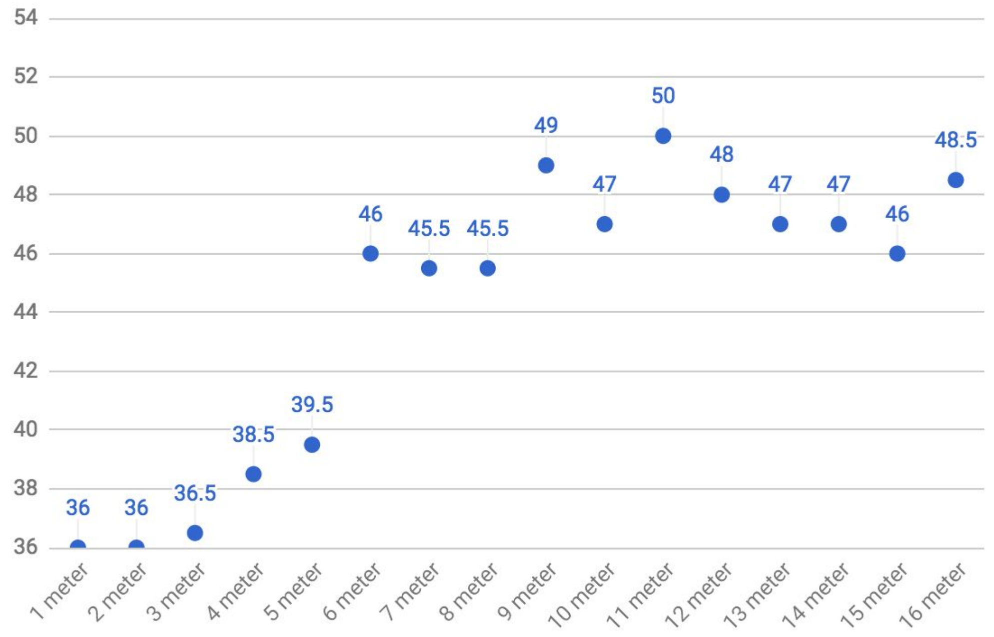
Then we did data collection on same system setting by meters in 17 meters since the pattern is limited in 6 blocks (16.5meters) due to the previous graphs. And below are graphs for both XBee S1 and XBee S1 Pro for 17 meters RSSI measurement by meters (Minimum Value):



Distance	1 meter	2 meters	3 meters	4 meters	5 meters	6 meters	7 meters	8 meters	9 meters	10 meters	11 meters	12 meters	13 meters	14 meters	15 meters	16 meters	17 meters
Standard Deviation	0.4215253442	0.6308903224	0.7446134249	0.4442825741	0.5166028756	0.6342396132	0.6231753025	0.6818883717	0.7991874404	0.8786144074	0.8223570918	0.870742264	1.14841453	1.153813378	1.27034294	1.10277001	0.8958472933

According to the graph, 17-meter distance is measurable. But the signal strength still drops at 5 meter. And the signal strength doesn’t change much from 9 meters to 14 meters. All these factors can interfere the system.

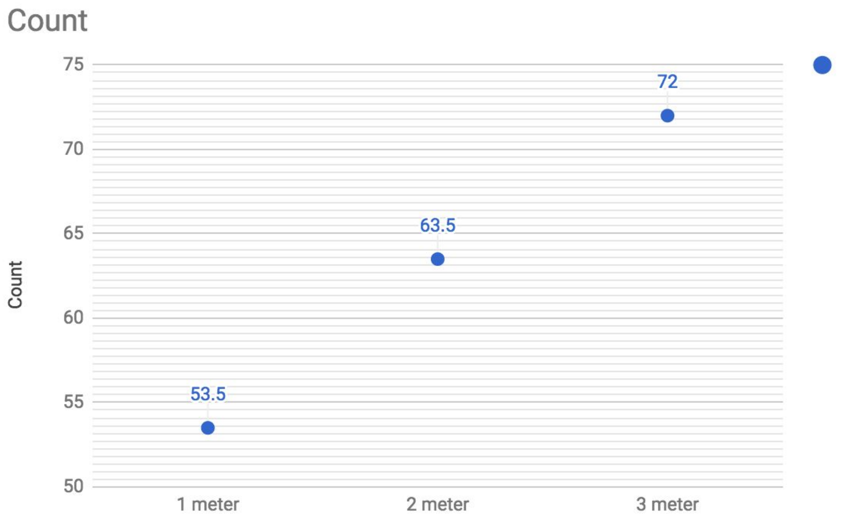
And below are graphs for both XBee S1 and XBee S1 Pro for 17 meters RSSI measurement by meters (Minimum Value):



According to both graphs shown above, XBee S1 Pro doesn't seems like a proper choose for accurate distance measurement.

Ground Data Collection(DayTime_Nocloud)

As Li, Ya and Guoshao said, RSSI shows the best pattern when XBee devices are 1.2 meters away from the ground (2014). And all experiment happened above were on 1.2 meters height. To prove that, we also did RSSI collection when the devices were setting on the ground (Medium Value) when power level is lowest on XBee Pro:



And the result shows that the signal strength become much lower and the decreasing speed of signal strength became high. The signal could not transmitted properly when distance is among 3 meters.

Formula Training with Least Squares

Least Squares is used to solve nonlinear normal equations problem (Levenberg, 1944). In Li, Ya and Guoshao's previous research, their maximum distance measurement error was decreased from 0.42 to 0.4 by using least squares rather than using mean method (2014). To use least squares method, we have:

$$\overline{\mathrm{RSSI}_{d_i}} = A - 10n \lg d_i$$

$$\boldsymbol{HX} = \boldsymbol{B}$$

In the formula above, H and B were already measured in our data collection process. We can then estimate X which is a 2 x 1 matrix that contains A and n values:

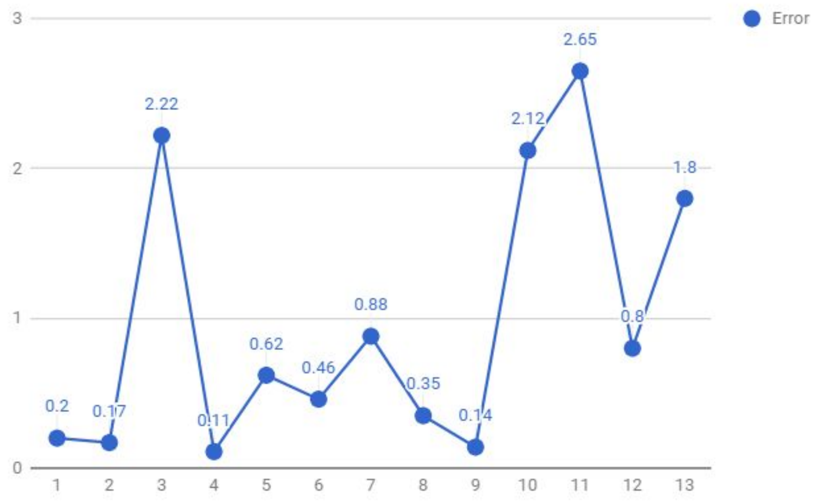
$$\boldsymbol{H} = \begin{bmatrix} 1 & -10\lg d_1 \\ 1 & -10\lg d_2 \\ \vdots & \vdots \\ 1 & -10\lg d_i \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \overline{\mathrm{RSSI}_{d_1}} \\ \overline{\mathrm{RSSI}_{d_2}} \\ \vdots \\ \overline{\mathrm{RSSI}_{d_i}} \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} A \\ n \end{bmatrix} 。$$

The formula used to calculate X is:

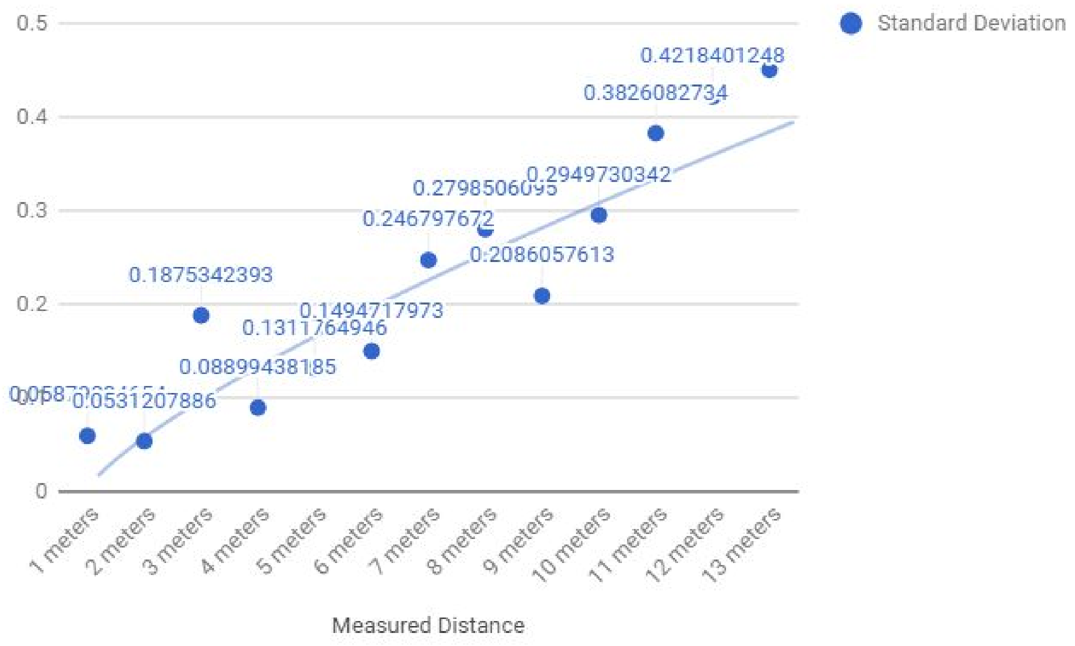
$$\hat{X} = (\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{B}$$

First Testing (Evening Dark Nocloud)

Since RSSI shows the best pattern under medium power level with XBee S1. Thus the least squares formula was trained with its data. The data collecting range for that data set is 1 to 17 meters. Below is the error average from 1 to 13 meters when distance is measured by meters, the sample size is 10:



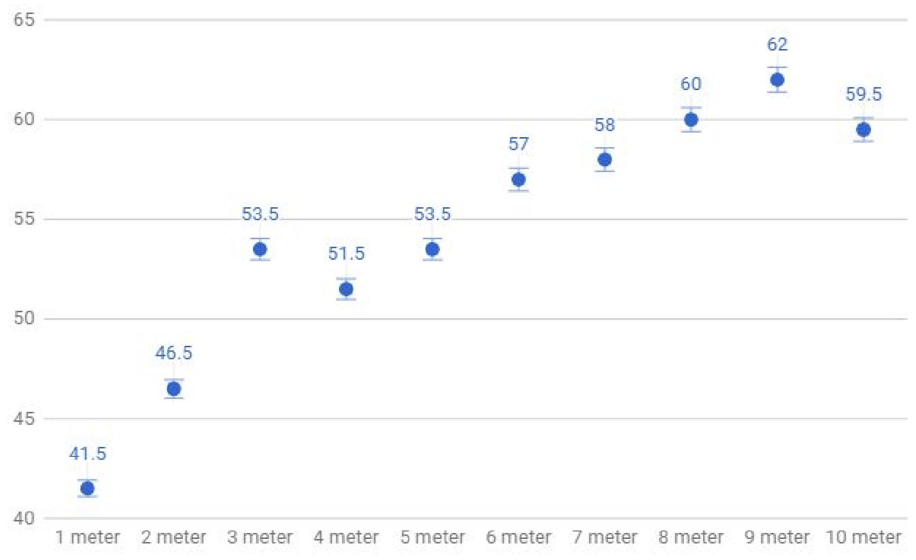
The standard deviation for the error measured on each distance increases when the distance becomes longer:



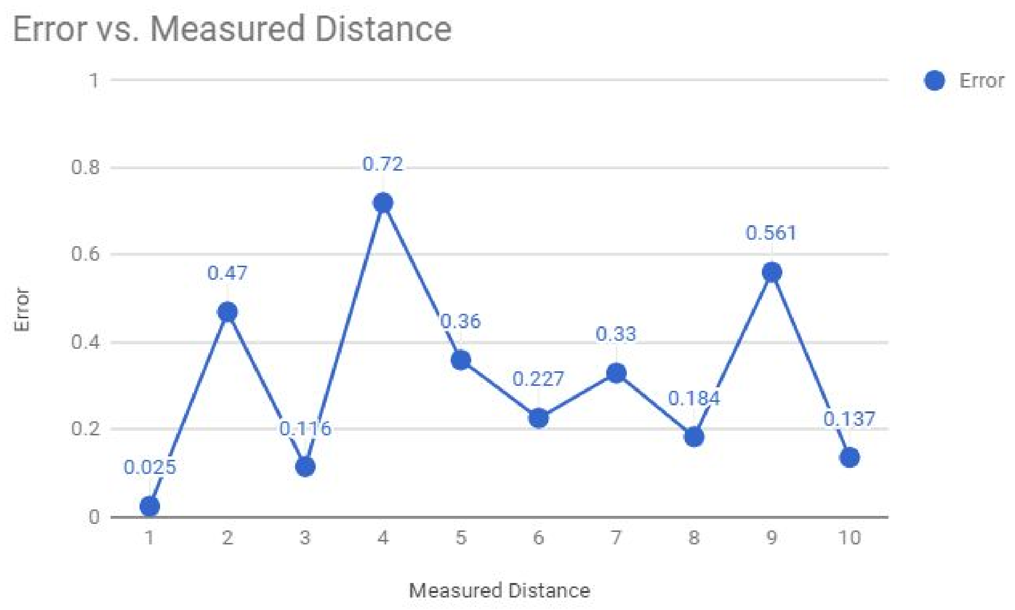
The testing longer than 13 meters was not able to happen due to signal weakness.

Second Testing(Daytime Nocloud)

After showing the last testing result with the client, our testing range was limited within 10 meters. To do the test, we were asked to train the formula within 10 meters data to do 10 meters testing. Since environment and weather condition determine the path loss value which is the most important variable in the formula. To leave the variable unchanged as possible, training data collecting, formula training and accuracy testing were happening in the same day. Below is the RSSI training data we collected with an average standard deviation of 0.37.



The average error for 1 to 10 meters distance measurement is shown below with sample size of 10, the total average is 0.313 when the maximum and minimum errors are 0.92 and 0.01.

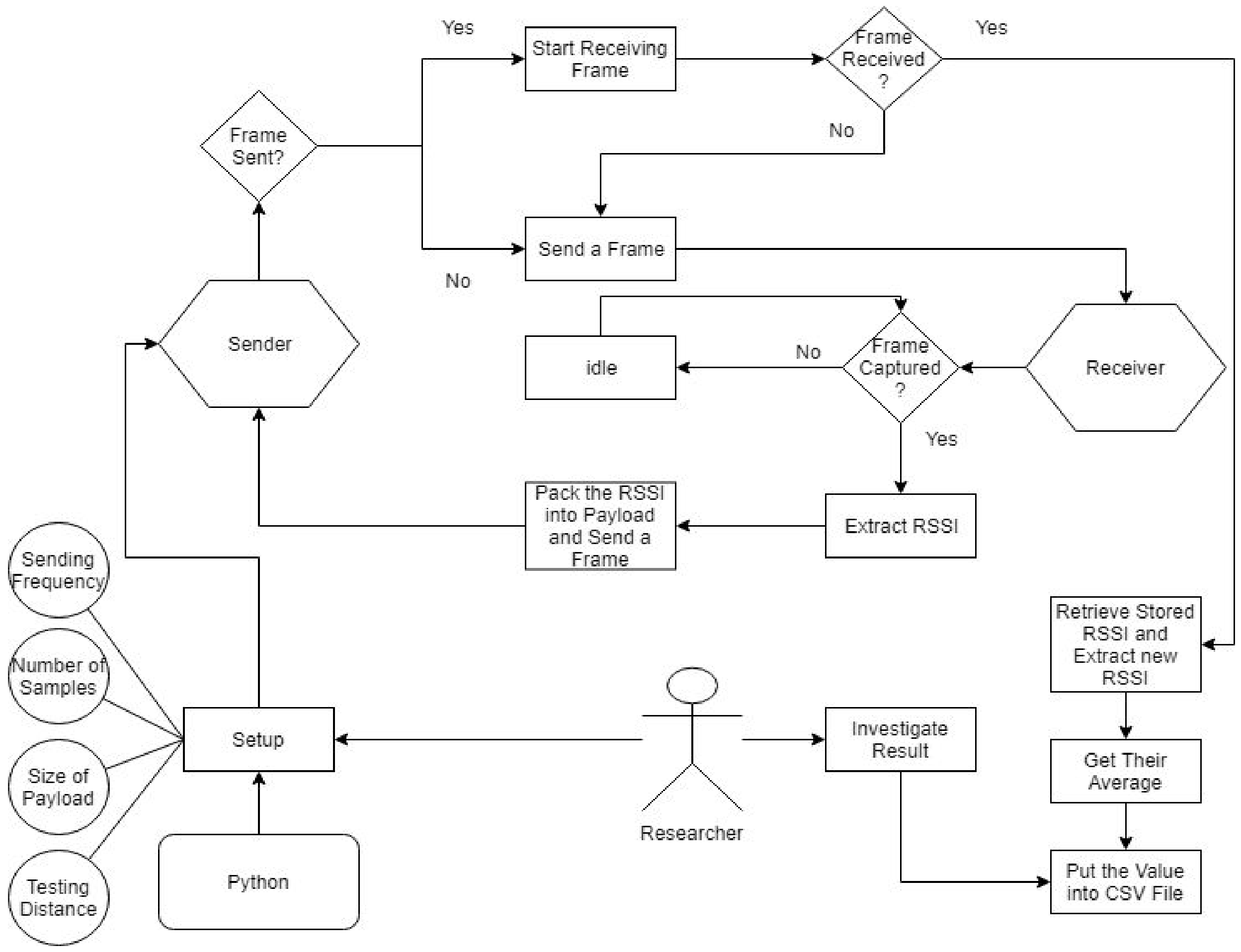


RSSI Testing Conclusion

Our RSSI measurement started from indoor environment covered by multiple electronics signals such as Wi-Fi signal. The original sending frequency was unable to be determined and the system was tatty. By iteratively delivering the system to our client and gathering feedback and updating the system incrementally, it has achieved cycle data collection, frame sending frequency and payload size modification. By measuring RSSI data based on technologies from current research papers, we did data collection in different power level, frame setting and environment. Finally we found the XBee S1 is the best choice for distance measurement when using ZigBee Protocol for point to point distance measurement. However, the testing range is limited into 10 meters due to unknown influencing factors.

Design

Since the RSSI is mostly affected by environment and its value is not changed linearly when distance is changed, the result will be more accurate and efficiency if we get both sender and receiver's RSSI average. And the signal might be lost due to environmental issue, thus both sender and receiver need to keep sending and receiving frames and avoid signal conflict. In other word, only one frame should be transmitted in the air at same time, otherwise the transmission will be frozen due to our experiment. To achieve that, we followed below structure:



Console (Python 2.7)

To achieve user input and data write to file functionalities, we use python 2.7 to connect to the Serial port of sender's Arduino board. To build the connection, both Arduino board and Python 2.7 communication rate were set to the same value 9600 as below:

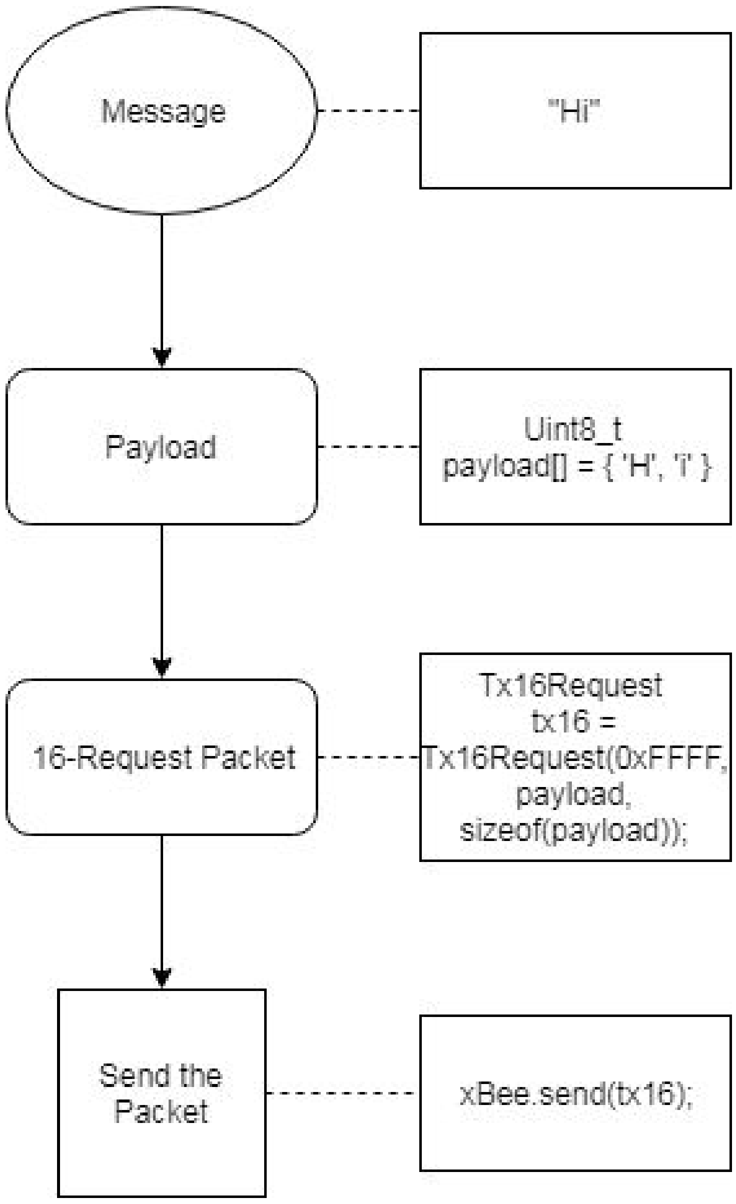
Serial.begin(9600); //Set Arduino serial data transmission rate to 9600

Ard = serial.Serial(COMP,9600) //Create serial port to communicate with Arduino port with same rate 9600(Library 'serial' is imported)

16Rx vs 64Rx (How the message is transferred)

To pack up the message to the sending frame, it will be added to an array which each char is represented as a byte. The array contains those bytes is called payload, and it will be captured by a 16 request packet and transferred to other devices. 16 request packet contains the information of 16-bit sender/source information.

There is also another class of request packet which contains 64-bit of sender/source information. However, our system only requires point-to-point signal transmission thus 16 bits are enough to contain the address information. Below is a diagram showing how the message is packaged and sent.


Message
"Hi"
Payload
Uint8_t payload[] = { 'H', 'i' }
16-Request Packet
Tx16Request tx16 = Tx16Request(0xFFFF, payload, sizeof(payload));
Send the Packet
xBee.send(tx16);

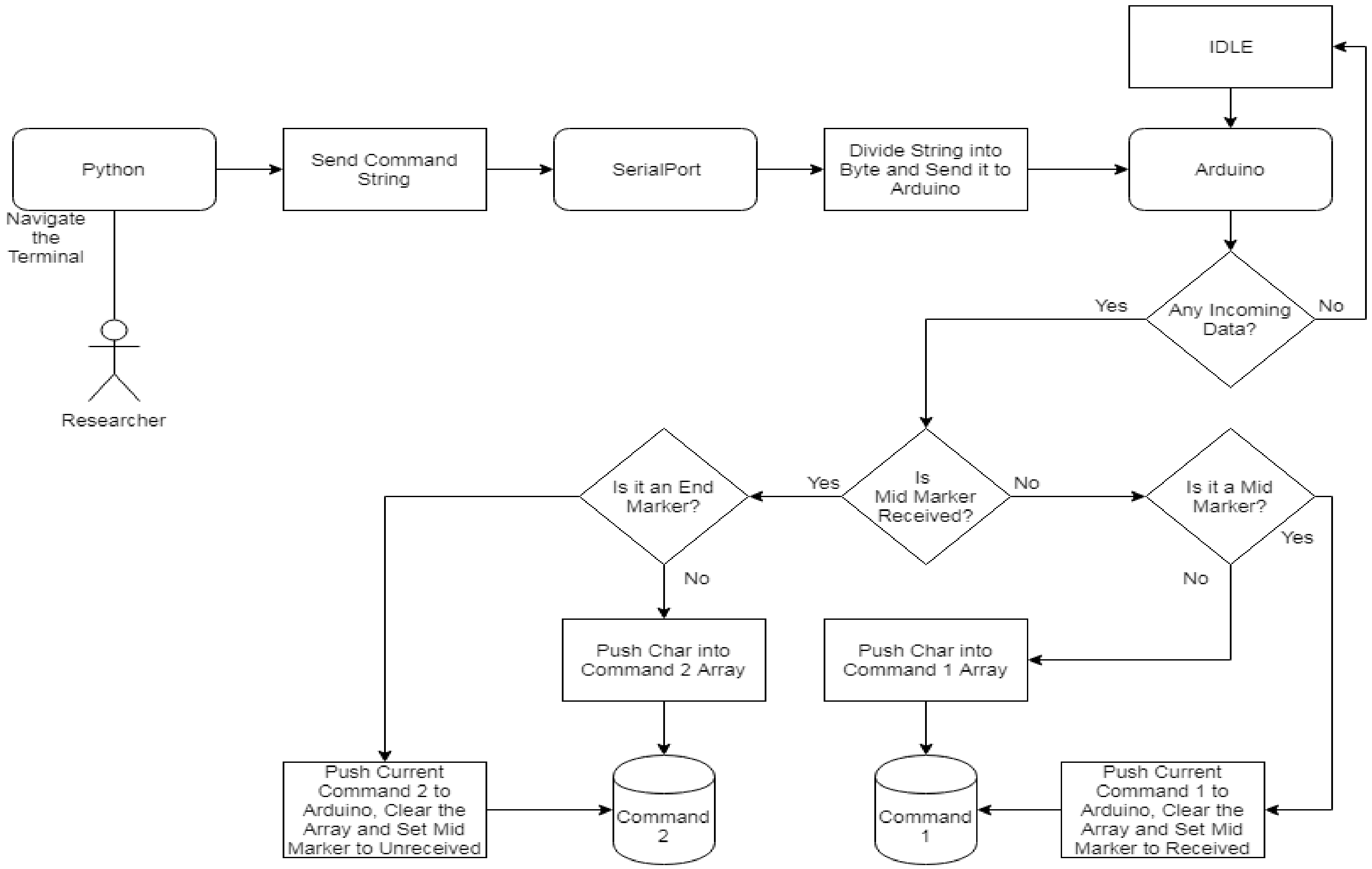
Sender

Meanwhile, Arduino board won't send any frame until it captures the command from Python. We run code below to keep it on waiting:

While (!Serial.available()){;} //When there is not message being transferred from Serial, keep the system into idle.

In the loop section, two main functions will be kept running, which are recvWithMarker() and gotNewData().

As Sustek, Marcanik and Urednicek say, the transmission from the serial buffer is read as one byte(2017). Thus the command sent from python to arduino will be read one byte by one byte as char. As we are receiving multiple sentences from the python terminal, a sentence containing multiple tasks need to be stored and separated into different sentence in the Arduino side. Thus we followed below structure in recvWithMarker function:


IDLE
Python
Send Command String
SerialPort
Divide String into Byte and Send it to Arduino
Arduino
Navigate the Terminal
Researcher
Any Incoming Data?
Yes
No
Is Mid Marker Received?
Yes
No
Is it an End Marker?
No
Is it a Mid Marker?
Yes
No
Push Char into Command 2 Array
Push Char into Command 1 Array
Push Current Command 2 to Arduino, Clear the Array and Set Mid Marker to Unreceived
Command 2
Command 1
Push Current Command 1 to Arduino, Clear the Array and Set Mid Marker to Received

Meanwhile, the two commands which are size of payload and sending frequency will be transferred into integer to Arduino for further progress.

Payload Size:

Int siz = atoi(Command1); //Create an integer that contains the information from Command.

Uint8_t payload[siz]; //Create the payload with designated size of bytes and put it into the frame.

payload[0] = sizeof(payload); //The payload shall contain the size of the payload, thus the receiver can send back a frame with same size of payload for a balence transmission.

Tx16Request tx16 = Tx16Request(0xFFFF, payload, sizeof(payload)); //This class represents an RX (Receive) 16 Request packet. Packet is built using the parameters of the constructor or providing a valid API payload. By setting the destination to 0xFFFF, the packet will be sent to every XBee in range with the same channel and ID.

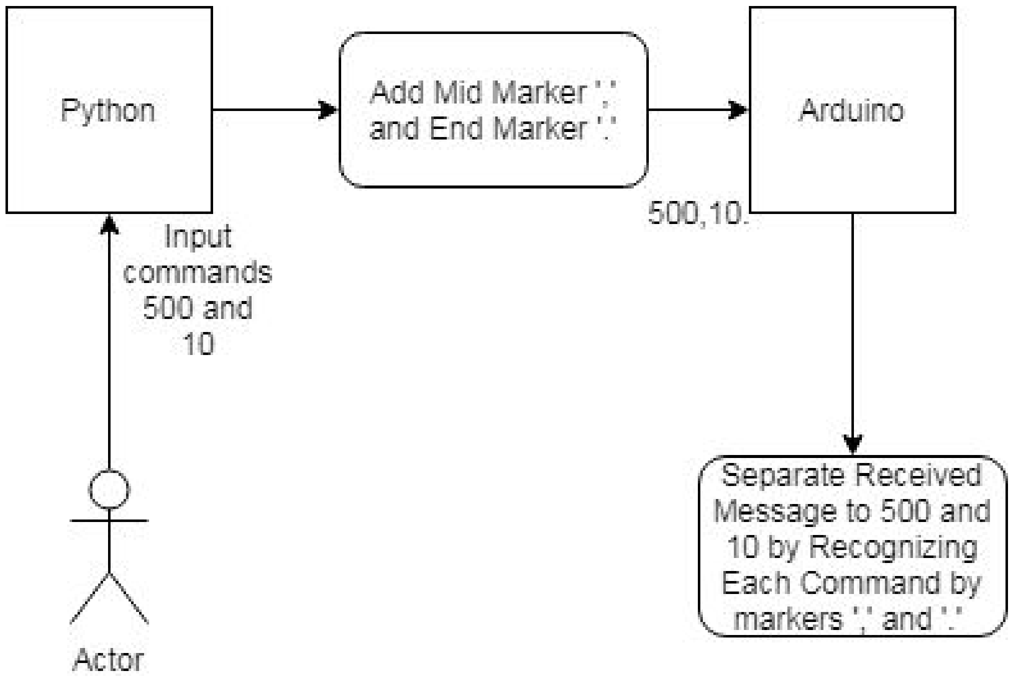
Sending Frequency:

To deter the signal sending frequency, we set time delay for every successful frame sending to set the system to idle.

Int count = atoi(Command2); //Create an integer that contains the information from Command

delay(count); //Set the system to idle for count milliseconds

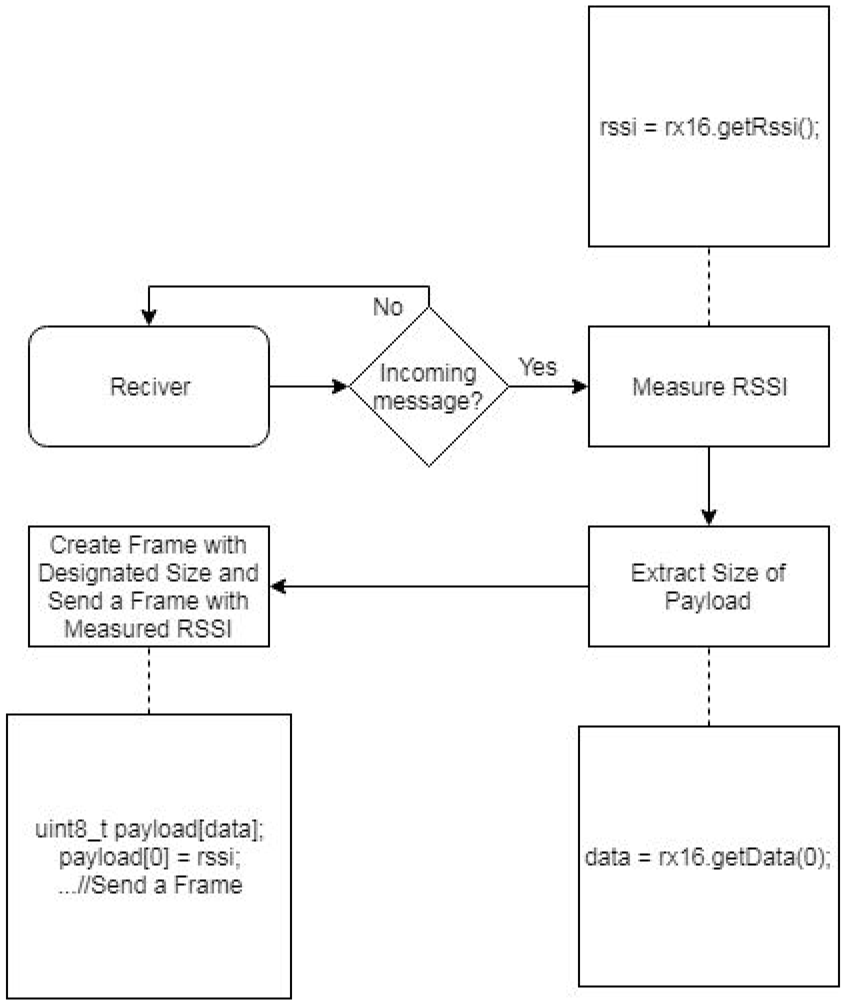
Below diagram shows an example on transmitting sending frequency of 500 every milliseconds and set the payload size to 10 bytes.


Python
Add Mid Marker ',' and End Marker '.'
Arduino
500,10.
Input commands 500 and 10
Actor
Separate Received Message to 500 and 10 by Recognizing Each Command by markers ',' and '.'

Receiver

The responsibility for Receiver is quite simpler than Sender since it’s only used to send a frame back to the Sender with designated frame size containing RSSI. To do that, the Receiver keeps reading incoming package every 100 million seconds which means the sending frequency cannot be above than this value.

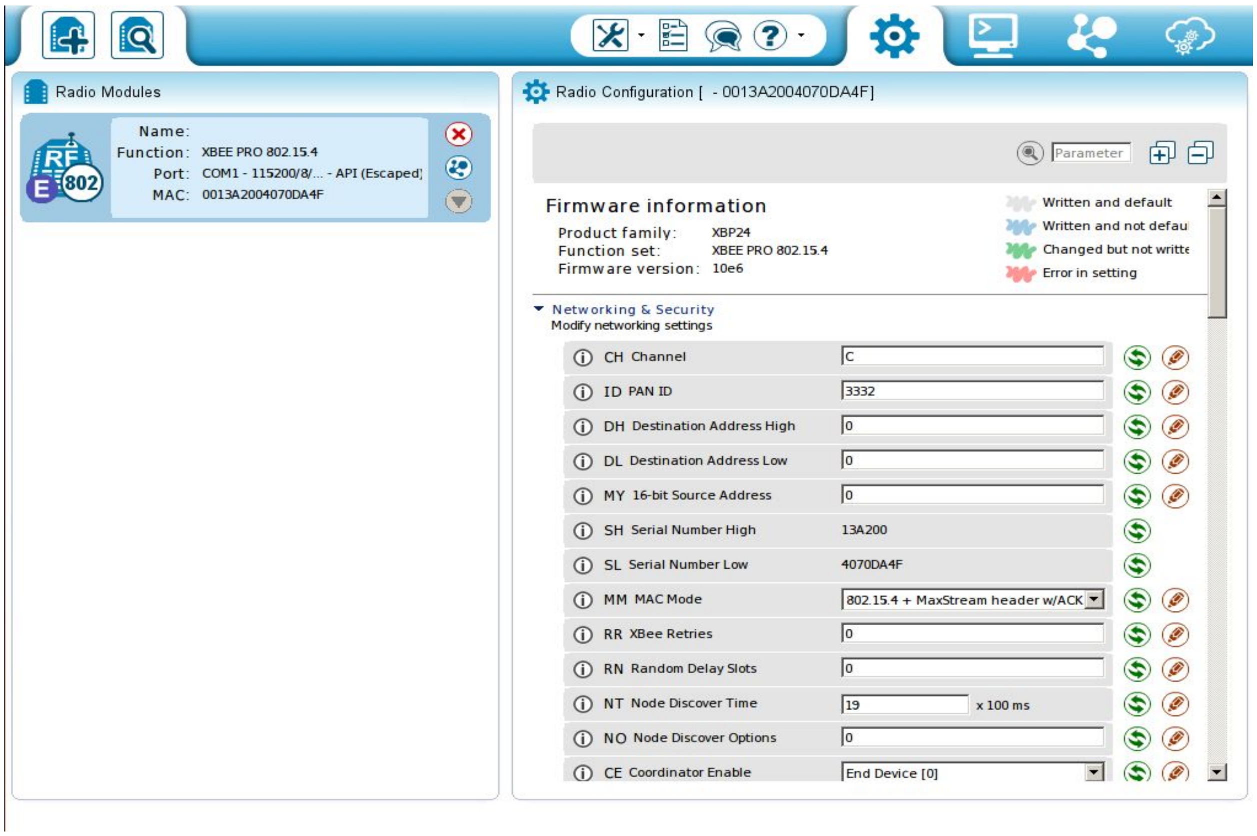
When xBee get response and the received packet has an apiID of RX_16_Response which is the one we send from Sender, the RSSI and payload size will be measured and extracted. Meanwhile, a new payload will be created with designated size and the RSSI which is just measured will be sent back to Sender. Below diagram shows what happens in receiver.



Implementation

XBee Handshaking

To implement the code by using the structure designed above. We firstly configured two sets of XBee modules XBee Series 1 and XBee Series 1 Pro with XCTU.


Radio Modules

Name:
Function: XBEE PRO 802.15.4
Port: COM1 - 115200/8/... - API (Escaped)
MAC: 0013A2004070DA4F

Radio Configuration [- 0013A2004070DA4F]

Firmware information
Product family: XBP24
Function set: XBEE PRO 802.15.4
Firmware version: 10e6

Written and default
Written and not defau
Changed but not writte
Error in setting

Networking & Security
Modify networking settings

CH Channel	C
ID PAN ID	3332
DH Destination Address High	0
DL Destination Address Low	0
MY 16-bit Source Address	0
SH Serial Number High	13A200
SL Serial Number Low	4070DA4F
MM MAC Mode	802.15.4 + MaxStream header w/ACK
RR XBee Retries	0
RN Random Delay Slots	0
NT Node Discover Time	19 x 100 ms
NO Node Discover Options	0
CE Coordinator Enable	End Device [0]

The picture above shows the interface for XCTU when XBee module is scanned and found in PC. As shown in the right area, there are mainly 5 features to configure in order to achieve the radio frequency communication:

- CH Channel: Defines the frequency to use to communicate. This must be the same for all radios on your network.
- ID PAN ID: Defines the network that a radio will attach to. This must be the same for all radios on your network.
- DH Destination Address High/Low: Defines the destination address to transmit the data to.
- MY 16-bit Source Address: These values should be unique to each XBee in a network as an address for each XBee in your network. The MY address can be any value between 0x0000 and 0xFFFF.

*https://www.digi.com/resources/documentation/Digidocs/90001456-13/tasks/t_wk_configure_xbees.htm

The destination address defines which XBee your source XBee is talking to. There are actually two values used to set the destination: destination high (DH) and destination low (DL). You can use that pair of values in one of two ways to set your XBee's mate:

1. Leave DH set to 0, and set DL to the MY address of the receiving XBee.
2. Set DH to the Serial Number High (SH) and DL to the Serial Number Low (SL) of your destination XBee.

Either method works, but the former – setting DH to 0 and DL to the destination's MY address – is usually easier.

*https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu

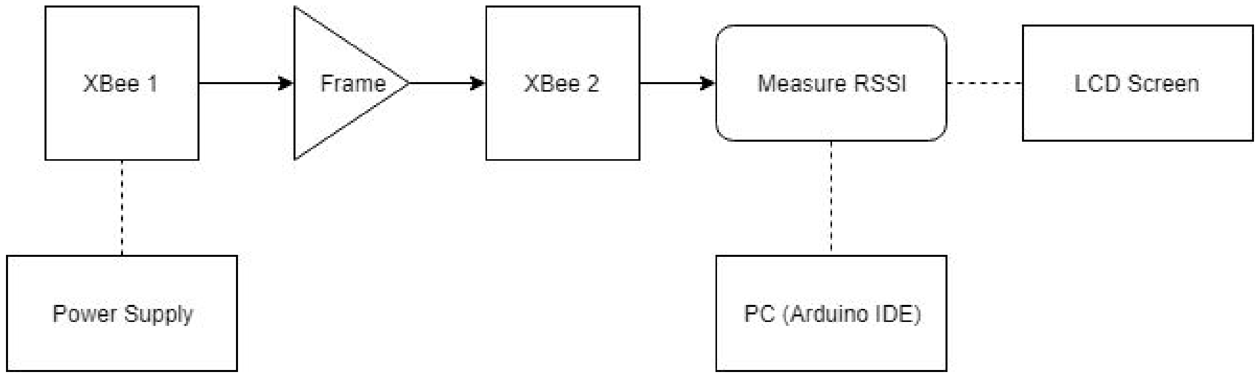
Thus our development team used the first option which is a convenient approach for point-to-point communication by using XBee Series 1. The configuration setting is:

Setting	Acronym	XBee 1	XBee 2
Channel	CH	C	C
PAN ID	ID	4910	4910
Destination Address High	DH	0	0
Destination Address Low	DL	1	0
16-bit Source Address	MY	0	1

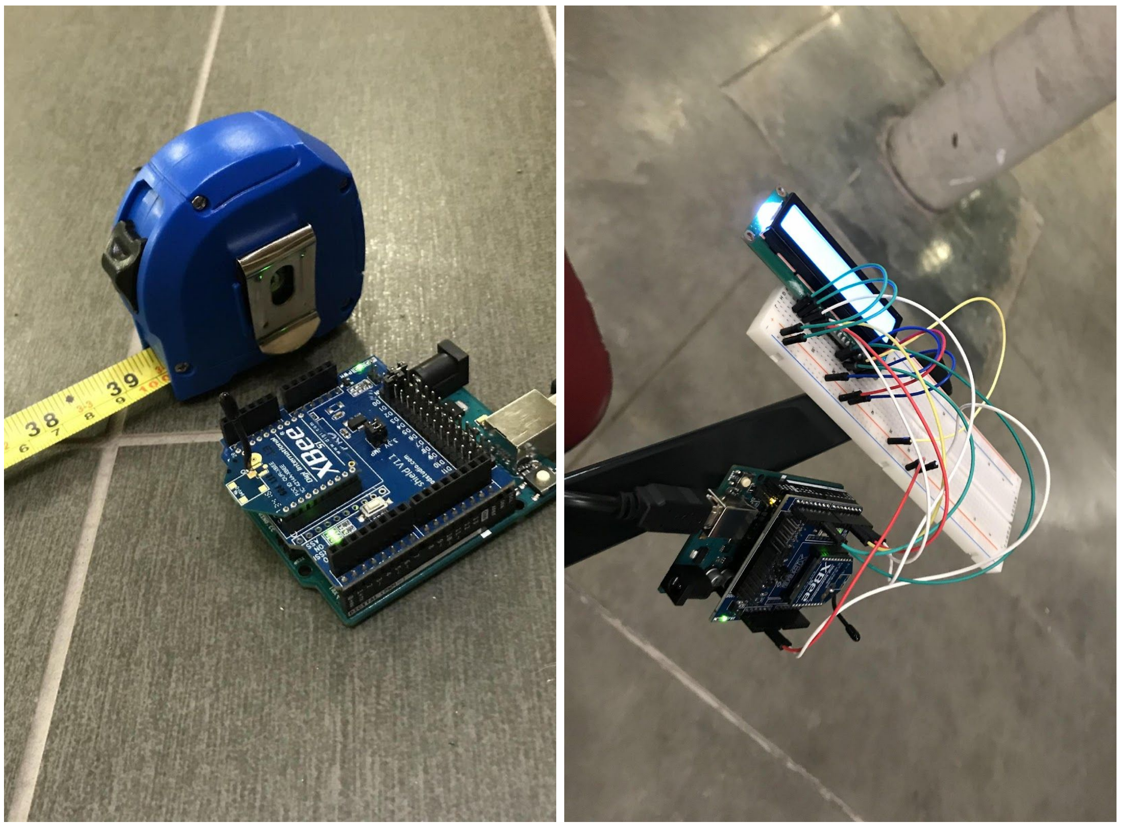
By using this setting, those two XBees can communicate properly through both PC and Arduino. So a few experiment happened after this.

Prototyping

Before going further research or implementation. We made an early version of our system as a prototype to achieve basic function and did a data gathering of indoor environment. Below structure shows how our system works in a prototype state.



As the pictures showing below, the sender (left) keeps sending frames with frequency and payload size specified from Arduino code. Receiver is connected to a PC and an LCD screen thus the researcher can monitor the data gathering process when data is being transferred. When the researcher collects enough data, we can pull out the power supply of sender to force stop the data transmission and then copy and paste those collected data into CSV file.



After talking with our supervisor and client, we summarized cons and pros of the prototype system.

Advantages:

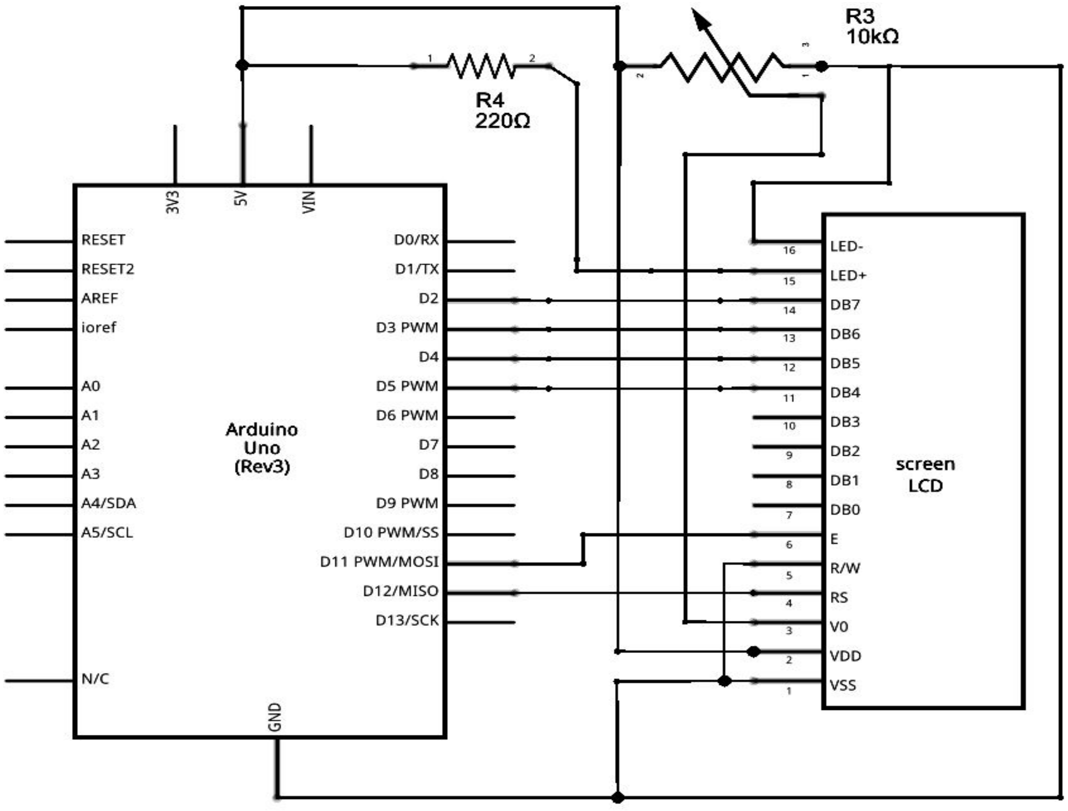
- Traffic monitoring with LCD

Disadvantages:

- Immutable signal sending frequency, payload size and amount of sample
- Hard to collect data
- LCD is hard to carry with wires while moving

To build the LCD properly and display RSSI on it, we used following code and circuits blueprint:

```
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);// With the arduino pin it is connected to
lcd.begin(16, 2); // Set up the LCD's number of columns and rows
lcd.clear(); //Clear the previous number when every loop happens
lcd.print(RSSI); //Print RSSI when every frame is received properly
```

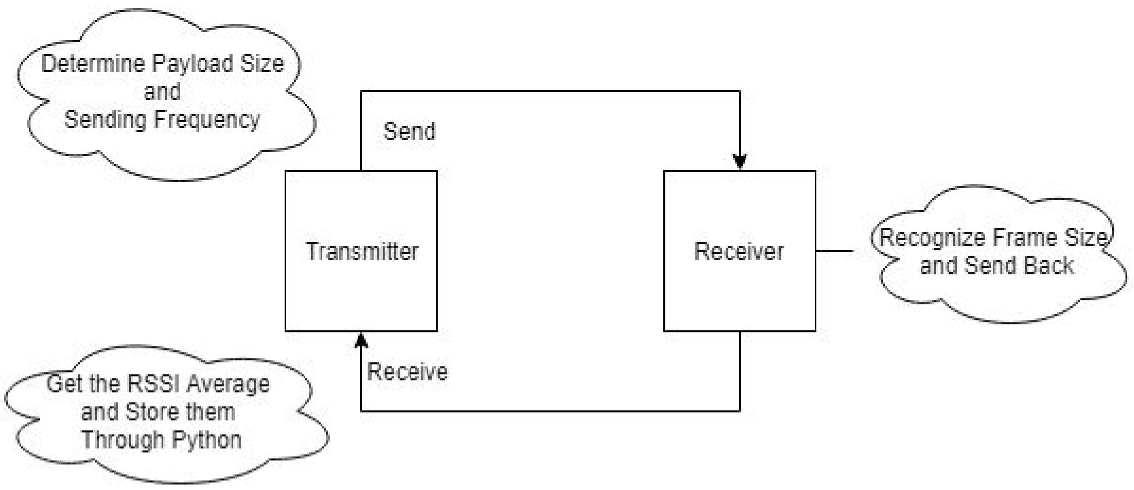


Evaluation

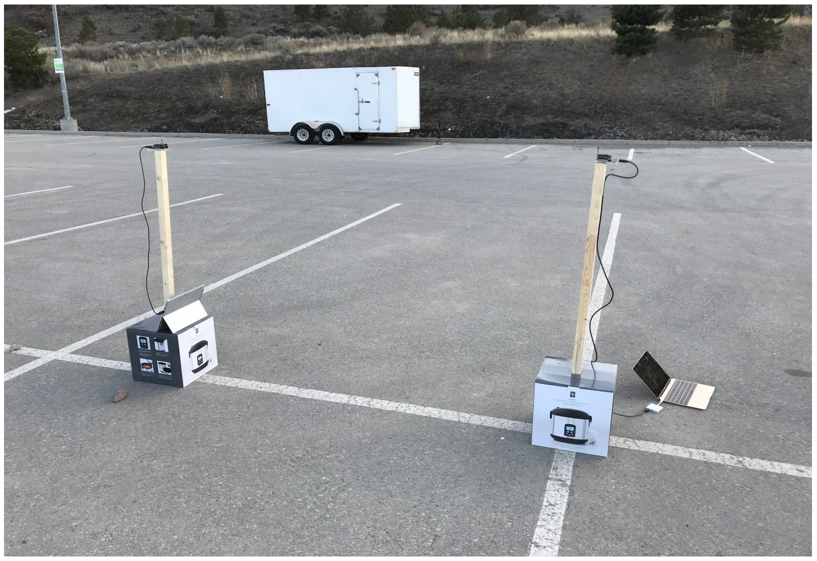
Since the prototype system is not well organized nor accurate on data recording. We had a midterm discussion with the client and instructor to generator below improvement measures:

- Getting the average of sender and receiver RSSI for precise result.
- Sending longer package to see if the signal becomes stable.
- Using python as the terminal for better control and documentation creation.
- Outdoor environment testing might lead a better result.
- XBee Series 1 Pro has better testing range than current device (XBee Series 1)

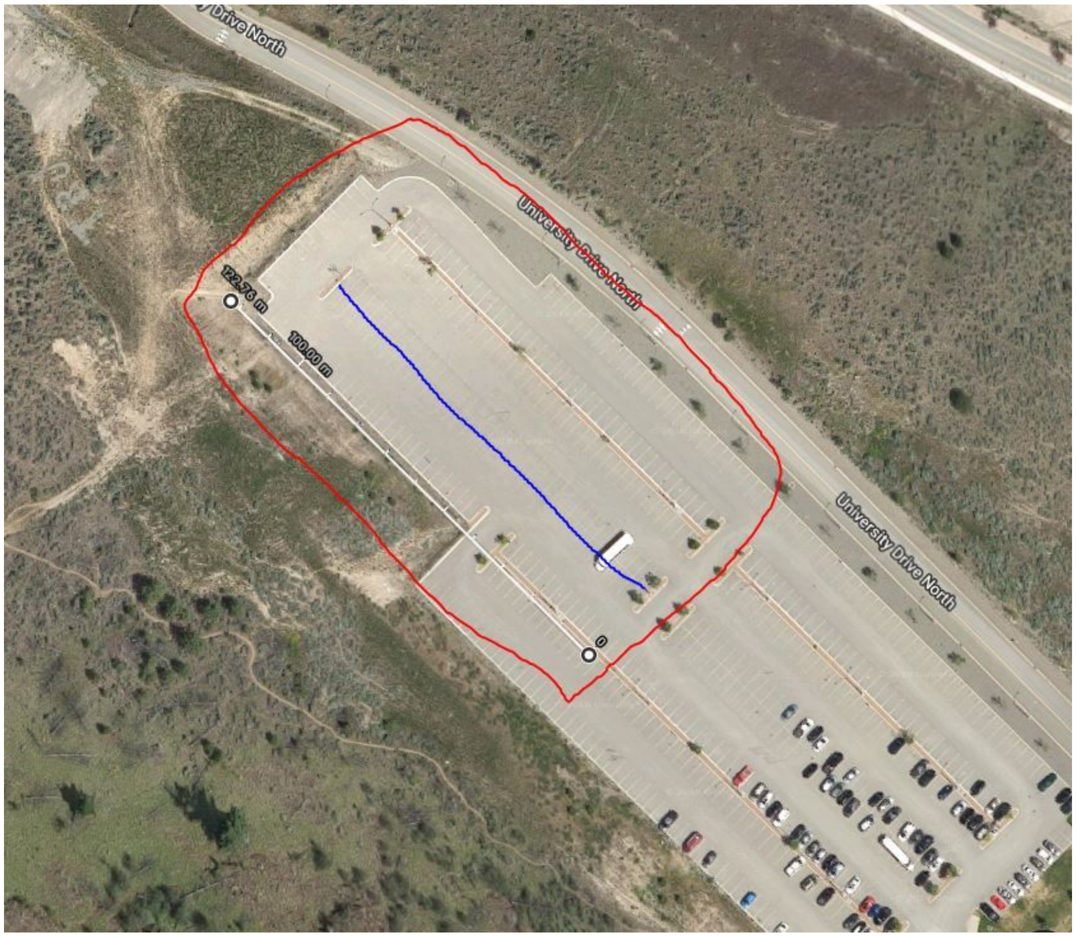
In this version, signals transfer in a cycle, which means the signal sent from transmitter will come back with both transmitter and receiver information. And next sending signal won't be triggered until the first frame is back. The structure is shown below, **see design part for more details at page 27**:


Determine Payload Size and Sending Frequency
Send
Transmitter
Receiver
Recognize Frame Size and Send Back
Get the RSSI Average and Store them Through Python
Receive

To ensure two objects are exactly away from the ground for 1.2 meters, our instructor provided us two sticks to support the XBee devices. Finally we combined the sticks with two foundations to build two XBee towers:



The testing field is a university parking area which has a big flat open area. The size of this field is 123 x 55 meters. The distance were measured through the blue line.



Flat Open Ground

Distance were mostly measured by ruler. However, when the distance became longer, we used parking block as the land marker for convenient measurement. Each block's width is 2.75 meters.

In the evaluated system, RSSI were measured in different power levels for both XBee S1 and XBee S1 Pro. **See more details in testing part on page 17.**

Code Structure

Environment: C language

- List of files

Name	**Description**	**List of function**
Rx_Gather.ino	**Gathering Xbee Receiver Rssi and sending Rssi value back to Transmitter**	
Tx_Gather.ino	**Xbee Transmitter**	**recvWithMarker(); gotNewData();**
Tx_Tester_Meters.ino	**Xbee transmitter convert average of Rssi values to meters**	

File name: Rx_Gather.ino

Library: # include <Xbee.h> #include<SoftwareSerial.h> #include <Printers.h>

List of global variables and constants

Name	Data type	Description
Unit_8 rssi=0	Unsigned only 8 bits integer	Define the getting rssi value type
Int data	integer	Define the getting data type

Additional description:

rssi = rx16.getRssi(); // Get Rssi value

XBee xbee2=XBee(); // Creat a new object

tx16 = Tx16Request(0xFFFF, payload, sizeof(payload));

xbee2.send(tx16); // Sending the rssi value back to the sender

File name: Tx_Gather.ino

Library: # include <Xbee.h> #include<SoftwareSerial.h> #include <Printers.h>

List of global variables and constants

Name	Data type	Description
unit_8 rssi=0	Unsigned only 8 bits integer	Define the getting rssi value type
float data	float	Define the average of rssi value
boolean sent =false;	boolean data type	Check and send a frame
const byte numChars = 32;	const byte	A constant value of array size
char receivedChars[numChars];	char	A char array to store the first input data from python
char receivedChars2[numChars]	char	A char array to store another input data from python
boolean newData = false;	boolean data type	Check the received char
boolean midData = false;	boolean data type	Check another received char
int count=100;	int	Initialize the delay time in milliseconds
int siz=1;	int	Initialize the payload size

List of functions

Name	Description	Parameters	Function that use

void recvWithMarker()	**Store the receiving python input in array and convert to string**	**static byte ndx = 0;
		static byte ndx2 = 0;
		char endMarker = '\n';
		char midMarker = ',';
		char rc;**
			Store the delay time input and the payload size input data from python

Void gotNewData()	**Convert the char array to integer**	**boolean newData count siz**	**Convert the char array to integer and use those integers to control delay time and payload size**

Additional description:

rssi = rx16.getRssi(); // get new rssi value

data = rx16.getData(0);// get the previous rssi value which is the receiver sending back package.

data = (rssi + data)/2; // get the average rssi value .

char endMarker = '\n'; // The end char marker symbol

char midMarker = ',';   // Use for separate received python input

delay(count); // control delay time

uint8_t payload[siz]; // control payload size

File name: Tx_Texter_meters.ino

Library: # include <Xbee.h> #include<SoftwareSerial.h> #include <Printers.h>

List of global variables and constants

Name	Data type	Description
unit_8 rssi=0	**Unsigned only 8 bits integer**	**Define the getting rssi value type**
int data	**integer**	**Define the getting data type**
boolean sent=false	**boolean data type**	**Check and send a frame**
float m=100	**float**	**Initialize rssi value / this m uses for selecting most strong signal**
int count=1	**integer**	**Initialize rssi value group**

Additional description:

Serial.println(pow(10, (-39.918+m)/19.614)); // convert rssi value to meters by our trained algorithm

Python IDE Code

Environment: Python language

Library: import serial   import time   import struct   import csv

List of function:

Name	Description
write_csv(counter)	Save the result from python terminal to csv file

Additional description:

ard = serial.Serial('/dev/cu.usbmodem14111' ,9600) // Link python ide to arduino ide with device port name and number

Conclusion

After the signal is sent from the transmitter, the energy of the radio signal will be gradually attenuated. Our researchers uses the attenuation law of wireless signals to measure the distance between the transmitter node and the receiver node. We mainly analyzed the attenuation characteristics of the signal with distance from the measured number according to the data from the relatively stable, after taking the average as the location RSSI value, study the variation of RSSI with distance and establish a ranging module type. During the study, we found that electronic signal and noise will have a certain impact on the stable propagation of signals. Thus certain data collection happened and were selected to reduce the environment impact factors. Finally our system was determined as a close point to point distance measurement approach by using XBee S1 ZigBee protocol, and the testing range is 10 meters with average 0.33 meters error.

*Our conjecture about interference factors except noise and other signal:

Atmosphere: Data measurement result in same day and same distance was very different when it became snowing from sunny day. Signal is transferred in the air.

Obstacle/ wall: Instead of transferring directly only, the signal can also be reflected from any obstacle. Our researchers found the testing result became messy when the nodes were next to wall.

Reference

Abu Sulayman, Iman & Almalki, Sami & Soliman, M.S.. (2015). *Design and implementation of a reliable wireless real-time home automation system based on Arduino Uno single-board microcontroller*. 2015. 2760-2764.

Cheon, J., Hwang, H., Kim, D., & Jung, Y. (2016). *IEEE 802.15.4 ZigBee-Based Time-of-Arrival Estimation for Wireless Sensor Networks.* Sensors (Basel, Switzerland), 16(2), 203. http://doi.org/10.3390/s16020203

D'Ausilio, A. Behav Res (2012) 44: 305. https://doi.org/10.3758/s13428-011-0163-z

F. Barrau, B. Paille, E. Kussener and D. Goguenheim, "*Distance measurement using narrowband ZigBee devices*," 2014 23rd Wireless and Optical Communication Conference (WOCC), Newark, NJ, 2014, pp. 1-6.

doi: 10.1109/WOCC.2014.6839958

Faludi, R. (2010). *Building wireless sensor networks: with ZigBee, XBee, arduino, and processing*. " O'Reilly Media, Inc.".

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. Quarterly of applied mathematics, 2(2), 164-168.

Salleh, A., M. K. Ismail, N. R. Mohamad, MZ A. Abd Aziz, M. A. Othman, and M. H. Misran. "*Development of greenhouse monitoring using wireless sensor network through ZigBee technology*." International Journal of Engineering Science Invention 2, no. 7 (2013): 6-12.

Srbinovska, Mare et al. "*Localization Estimation System Using Measurement of Rssi Based on Zigbee Standard*." (2008).

S. Schwarzer, M. Vossiek, M. Pichler and A. Stelzer, "Precise distance measurement with IEEE 802.15.4 (ZigBee) devices," 2008 IEEE Radio and Wireless Symposium, Orlando, FL, 2008, pp. 779-782.

doi: 10.1109/RWS.2008.4463608

Sustek, M., Marcanik, M., & Urednicek, Z. (2017). *Using of Inputs and Outputs on Microcontrollers Raspberry and Arduino. International Journal of Applied Engineering Research*, 12(13), 3944-3949.

Verma, M. (2017). *WORKING, OPERATION AND TYPES OF ARDUINO MICROCONTROLLER*.

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY,

6 (6), 155-158. doi:10.5281/zenodo.805403

V. K. Sehgal, Nitin, D. S. Chauhan and R. Sharma, "*Smart wireless temperature data logger using IEEE 802.15.4/ZigBee protocol*," *TENCON 2008 - 2008 IEEE Region 10 Conference*, Hyderabad, 2008, pp. 1-6.

doi: 10.1109/TENCON.2008.4766744

Z. Jianwu and Z. Lu, "*Research on distance measurement based on RSSI of ZigBee*," *2009 ISECS International Colloquium on Computing, Communication, Control, and Management*, Sanya, 2009, pp. 210-212.

doi: 10.1109/CCCM.2009.5267883

Li Z, Ya L, Guoshao C. *Distance Measurement Model of RSSI Based on Zig Bee.* Ordnance Industry Automation. 2014;12:52-5.

https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu/configuring-networks

http://www.trossenrobotics.com/p/arduino-uno.aspx

https://www.arduino.cc/en/Guide/Environment

https://www.arduino.cc/en/Reference/SoftwareSerial

https://www.arduino.cc/reference/en/language/functions/communication/serial/

https://www.digi.com/resources/documentation/Digidocs/90001456-13/tasks/t_wk_configure_xbees.htm