SINGAPORE POLYTECHNIC

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

PROJECT NO. 15A155

MOUSE TRAP ALERT SYSTEM

SUBMITTED BY:

1321706 Ong Jia Chen

1321016 Lim Ting Wei

Project Supervisor: Mr Wong Chee Yong

YEAR 2015/2016

ACKNOWLEDGEMENT

We would like to express our appreciation to our supervisor, Mr Wong, who has been guiding us on our journey during the Final Year Project. He has been very helpful in teaching us, giving us a boost to move on when we are in tight situations. This project would not have been smooth-sailing without him, therefore we were very grateful that we have a caring and supportive supervisor.

We would also like to thank Mr Terence, for giving us feedback and suggestions to make this project more meaningful. He has been willing to help us in providing hardware components, creating casings that are suitable for our prototype, and giving us access to his database server.

In addition, a thank you to Mr Afendi, who is the first to know our prototype. He is always interested in solving problems in his company, PestBusters, and has given suggestions to us to further improve our prototype.

Last but not least,we would like to express our gratitude to PestBusters, who has given us a chance to make this project a successful one.

ABSTRACT

Purpose : The purpose of this project is to create an alert system that keeps track of mouse traps to tackle lack of manpower and improves the efficiency of managing traps in a specific area.

Brief Description : Using ZigBee for Personal Area Network, the system can communicate with the mouse traps within a certain range.

With the help of a user-friendly application, the user is able to monitor the traps and gets notified by SMS whenever a mouse trap is triggered.

Furthermore, users can view the status of the mouse traps from a database server via a second application.

Conclusion : The system is tested thoroughly and improved over a year to be as stable and reliable as possible. It is currently used in a site area via live monitoring. We hope that this system can be useful to manage the traps efficiently.

TABLE OF CONTENTS

INTRODUCTION

The Mouse Trap Alert System is a system that monitors multiple mouse traps in a specific area. When any one of the mouse traps is captured, the system will SMS to the staff to alert them that the mouse trap is triggered. The staff will proceed to the area to collect it easily since the location is written on the SMS message, giving them convenience and at the same time improves efficiency of monitoring. Furthermore, the staff can check the mousetrap status remotely via a second application.

Target Audience

Industrial companies that specialize in pest management especial rats

Background Information

With the lack of manpower, the staff has to go to the place where mousetraps are mounted. Every night, he has to check every mouse traps located in that area. This is because these traps are one-time only. It means that when the mouse is captured, the trap cannot be used again until the staff comes and collects it. However, this results to be inefficient as he would not know which mousetrap is ready to collect and there are tons of mousetraps. Here are some of the basic information:

- Minimum of 25 to 50 mouse traps are deployed in a large area, it is very hard to monitor all of them.
- Frequently sends staff to check all mouse traps 4 nights a week. It is not efficient as he may not find any captured mousetraps on that night.
- Pesticide is largely used in those affected areas.

Scope

This report covers the ideation, design thinking, hardware, software, features, implementation, problems encountered and conclusion of the project.

Purpose

The purpose of this project is to create an alert system that keeps track of mouse traps to tackle lack of manpower and improves the efficiency of managing traps in a specific area.

MAIN TEXT

2.1 CDIO

We started this project with the problem that the industrial we are working with had management problem when comes to manpower. Therefore, we are thinking feasible ideas that we believe that would improve their efficiency of their schedule on hand and have different ways which we can target this issue. However, what the industrial wanted to solve is the problem with rodents that cause the manpower problems, which now we can narrow down to just improve the efficiency of managing mouse traps.

As we research for more information, we found out that some areas singapore are infested with rodents, mostly cause by rats which the industrial are facing. Based on Mr Afendi, one of the employee had told us in one night, rats can be caught up more to 20+ just by using both spring traps and glue traps. Therefore, through existing product on the market, we come out with our ideation that would able us to solve our need statement:

- To increase efficiency of managing mouse traps with the help of the today's technology

This directed us to think of smart solution for ideation which able to solve their problem. We came out with 4 feasible ideas that would work on-site:

- Motion Tracking MouseTrap System
- IR sensors MouseTrap + IP Camera
- Attachable MouseTrap Phone Alert System
- Ultrasonic Pest Repeller

Motion Tracking MouseTrap System involves tracking down the path for rats within the area that the system is being deployed and able to capture them effectively. The employees would learn the rat behaviour, strategically set traps in their path and utilize the system more effectively.

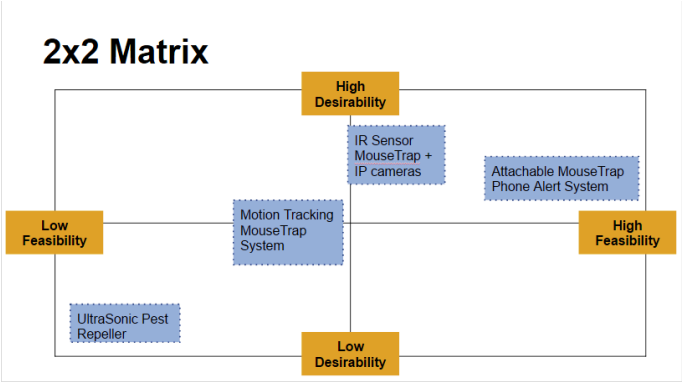
IR sensors MouseTrap + IP Camera involve capturing image of what is inside the trap when rodent walks in and triggered the IR Sensors, this will prevent any false alarm as when the trap is activated. If the the image does not show any rats in the traps they will not need to go empty trip down to the site to collect back the traps.

Attachable MouseTrap Phone Alert System AKA. MouseTrap Alert System involve sending alert to employees when rat has been caught in the trap, thus the employees would not need to go down to the site until alert has been send. It is attachable to any default string traps on market, developing of the cage itself is not a priority. Since employee are hold onto their personal phone, information can be access 24/7.

Ultrasonic Pest Repeller involves strategically deploy it on-site which employees with long experience can use the repeller to redirect the rats to a specific area where the spring traps

and glue traps are hidden. This system will irritate the rats to come out from their hiding spot, thus capturing rat in a single night is possible.

After much consideration as we compare the feasibility between each of the systems, we had chosen MouseTrap Alert System to be the best out of the 4 ideas.



Reason for this choice is because is attachable to default mousetrap, we do not know much about the size of the rats and how big should we design so that the rats would fit in. Therefore 1st and 2nd choice already out even IP cameras which has high desirability are not selected.

Each site has different hiding spot and hot spot for rats which most of the employees will not effectively redirect the rats to the cage or rather away from the site and infect another new site. Therefore, 4th choice is out.

Best about 3rd choice is the use of phone app to communicate with the system. Apps can be one of the rising usage for today's technology as most of us as from young to old are adapting to use apps everyday. Which we believe the employees as well will be able to learn to use the system app very quickly. Information can be updated through apps with syncing to the palm of the hands and can be access 24/7 anytime, anywhere.

We thought about the design thinking, ideal functions and solution and developed a two part system: Main System(Master) for management, Traps(Slave) for capturing and notifying.

Main prototype set includes:

- Arduino as our microcontroller. (Embedded System Design)
- App interface to easily identify how many traps are being deployed.
- Using Magnetic Reed Switch to send signal to the Arduino once the trap is triggered.
- Using Zigbee/Xbee PAN (Personal Area Network) technology to communicate between 2 Arduino boards wirelessly.

During the start of the project, Both of us have decided that Hardware will be done by JiaChen while Software Android Studio will be done by Ting Wei. Arduino Code will be done by both of us which correspond to what function we need for our system.

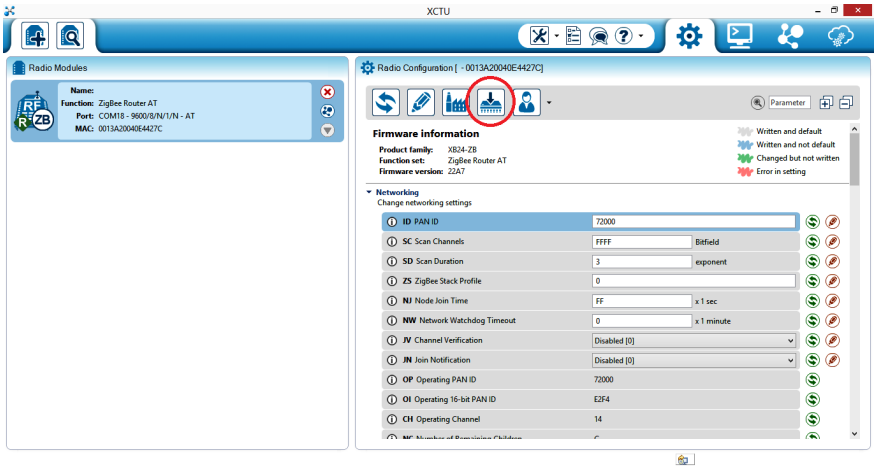
2.2 Hardware

Hardware development has been trial and error throughout, from changing of main component, Arduino UNO + Zigbee/Xbee Shield > Arduino fio v3 to the smallest detail, normal single core wire > rainbow wires so that it will not easily break and stuck in the Arduino Board.

For easy reading, I (Jia Chen) will be covering into 3 parts Setting Xbee Communication, Main System(master) and Trap(slave).

Xbee Communication

Using the Software X-CTU will allow me to see the Firmware information.



This user-friendly GUI interface also allows me to do:

- Configuration of Xbee
- Update of new Xbee firmware when there is a new patch from XCTU site
- Range Test between 2 Xbee from a distance.
- Test of data in data string or data packets on a timely basis.
- Number of data packets to send in a period of time.

At the very beginning, I updated the firmware of the Xbee to the newest because there might be bug in the old firmware.

For the configuration setting, most of it will stays in default and only a few will be changed:

- (ID) PAN ID: All Xbee from the same (Personal Area Network) PAN will be set to a same ID. For our case would be ID 72000
- (DH) Destination High: Since I am broadcasting the data to all traps, set the field to 0 for the upper 32 bits of the 64 bits register.
- (DL) Destination Low: Since I am broadcasting the data to all traps, set the field to FFFF for the lower 32 bits register.

Extra Setting for Trap(Slave) field:

- (SM) Sleep Mode: Set sleep mode to 1 to allow the xbee to sleep when necessary.

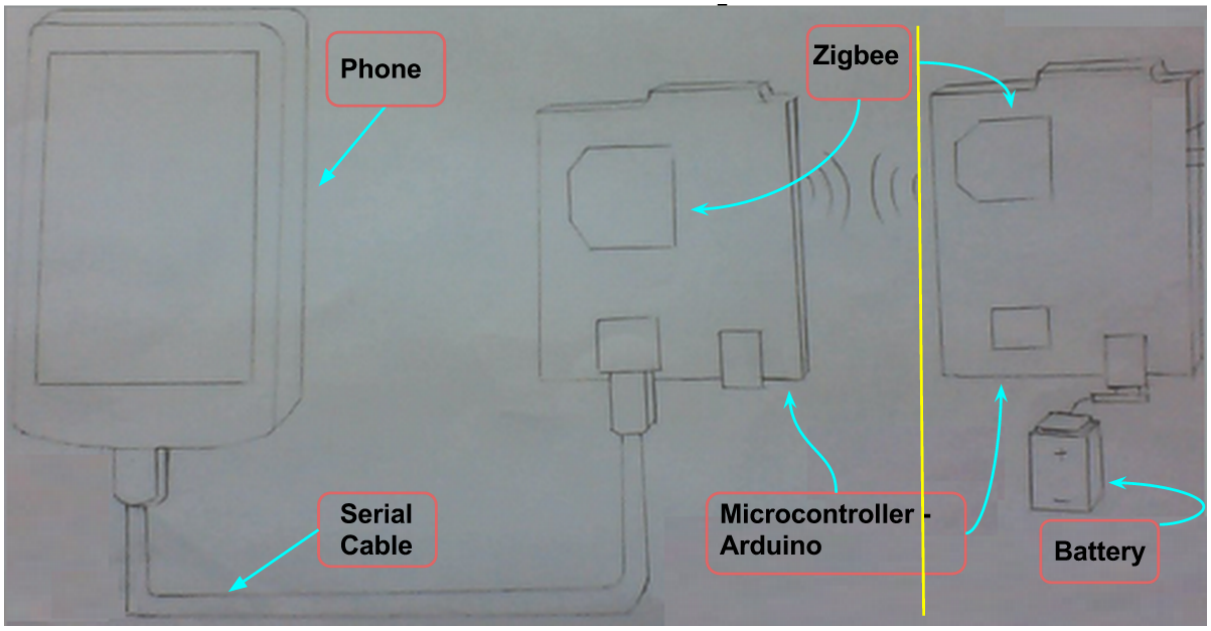
After all this is done, I plug two xbee to the computer and test the communication using the console terminal. Type a phrase on one side of the console, it will display it as blue but on the other console terminal, the same phrase will display as red, which means the data has been transferred successfully and no problem with the configuration.

Therefore, whatever happens to the data that did not send out would mean that error is at the coding of the arduino and android instead of the hardware configuration.

Now for each parts, I will state down the changes that we made from the first concept sketch to the final decision and finally the end-prototype.

<u>Main System(Master)</u>

Concept Sketch:



The yellow is the split, left side is the component for master and right side is the component for slave.

We started out thinking of something simple, efficient and most important, the budget for our prototype. Therefore 4 main components for master is a must to have in our prototype:

- Phone
- Arduino Uno
- ZigBee/Xbee
- USB cable for communication between the Arduino Uno with the Phone.

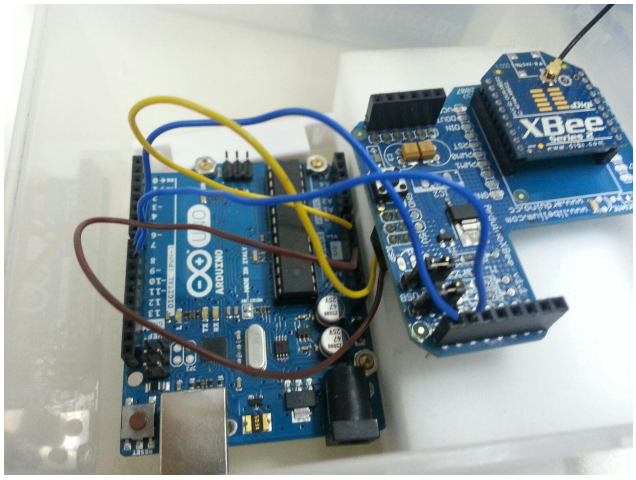
However, problems arises when I gotten hold of the components and research on it:

1. Zigbee/Xbee cannot plug directly onto the Arduino Uno and have to add cost to the budget by getting the Zigbee/Xbee shield to stacked onto the board.
2. Zigbee/Xbee cannot plug directly onto the Zigbee/Xbee as well since we have two data link coming back and forth.
    - Zigbee/Xbee is communicating through pin 0 and pin 1 (RX, TX) between phone through serial cable.
    - Zigbee/Xbee is communicating through pin 0 and pin 1 (RX, TX) between slave Zigbee/Xbee through wireless.

    Therefore, I have to pull copper wire from the pin 0 and 1 of the Zigbee/Xbee shield and plug onto another pair of pin port on the Arduino Board. Those new pair of pin port will need to code in and initialize that it is “RX, TX”, “RX” will set as INPUT and “TX” will set as OUTPUT and start the connection as serial1.begin instead of serial.begin as it is used by the serial cable.

3. Using the serial cable from the phone to the Arduino Uno Board to power up. This causes the phone to used up all its internal battery and the whole things would not worked anymore. Therefore, we try to find the y-cable that is capable to charge up the phone and power up the Arduino Uno Board at the same time.

This is our very first prototype:



The Copper wire is all tangled around which is already a failure, once cable drop out and that’s it, the whole system will fail. But for the time being, due to time constraints we will showcase to the industry.

After showcase, I am back to the drawing board and research on a more reliable Arduino Board. However, What we realized is that Zigbee/Xbee has the capability to send the data out without the help of Arduino fio V3 since the coding for master can be embed into java code which initially used C coding for communication. Hence, we save the cost of 1 Arduino Fio

Board and get the dongle which the Zigbee/Xbee will sit on just to communicate with other Zigbee/Xbee.

Now left the last solution for serial cable which allow charging the phone and power up the Arduino Board. We went out to sim lim square to get serial cable AKA Y-cable and seems that either those cable are just power cable (Only supply power, not both data sending and power supplying) or the power is sucked up by the Arduino Fio v3. Seems that I have to find it online and got to this site called valarm which claims to do the jobs that we needed to do, charges both the devices with data communication purposes

INDUSTRIAL IOT - MICRO USB HOST OTG Y-CABLE WITH MICRO USB POWER CHARGING FOR SAMSUNG PHONES

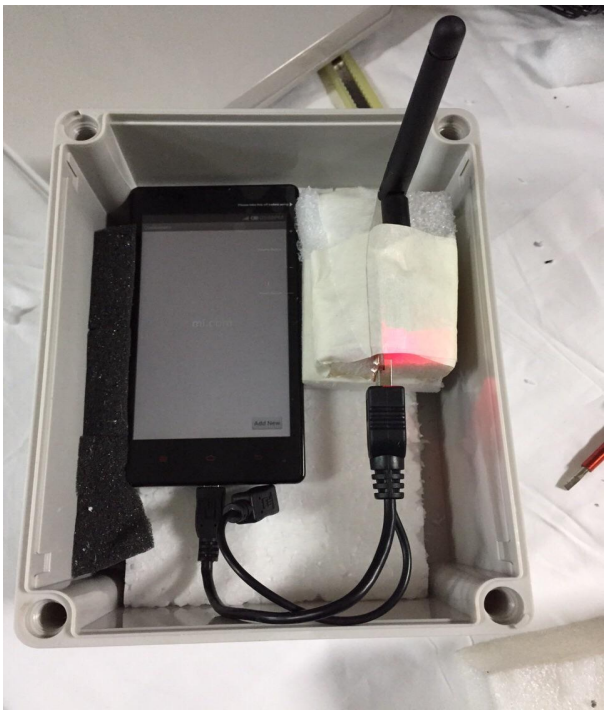
19^{00}

Add to Cart

Note: This cable is for **Industrial IoT applications with **Valarm Tools Cloud**. We recommend using GSM, WiFi, or ethernet connector devices / sensors hubs since they're more reliable than Androids for Industrial IoT applications! Have a look at Valarm's **Customer Stories** page for more on how these remote sensor monitoring and telemetry solutions are deployed in various industries like natural resources, agriculture, oil & gas, governments, petroleum, and water + fluids. Also see how all of this works together on our page **Web Dashboards for Remote Monitoring, Telemetry, Sensors & Industrial IoT**.

We now have a new, improved and completely custom version of this cable manufactured just for us! We had to buy 50,000 custom resistors in order to have this cable manufactured. **No other** "Samsung compatible" OTG Y-cable currently available has been manufactured to this specification. You think the cable from Amazon or Newegg will work? Wrong. Ours is the only one that works, so save your money and get it here. Otherwise it won't work. :)

Now all the problem are solved, this is the final prototype of our master



(Figure.5)
Item consists:

- Android Phone
- Zigbee/Xbee Dongle
- Zigbee/Xbee 2mW RPSMA - Series 2

- RP-SMA Antenna
- VAlarm OTG Y-Cable

Since we are doing an industrial project, the Casing for the master is provided by the company. The casing itself is rigid and it is water-resistance, good for deployment on trial.

Trap(slave)

Slave, one of the hardest hardware to do since the project need to include a lot of external components and link it all up to the Arduino Board.

Concept Sketch:
In figure.1, we try to simplify the whole slave, which the main component of the it consist of Arduino Uno, Xbee and a battery holder to power the slave. However, Xbee does not sit directly onto the Arduino Uno Board, so a shield it bought to solve the problem.

Now the limitation for slave are stated below:

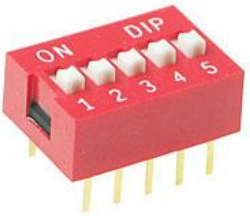
1. The shield is stacked onto the whole Arduino Uno Board and external component are not able to access those pins on the Board itself.

2. Components like resistors and capacitor needed a Cooper Board to be able to solder. Which I need to think of how to connect up Cooper Board to the Arduino Board

First limitation is really can't be avoided since the hardware is design it this way and therefore I will first plug whatever cooper wire to the pin hole I needed and just bent it to 90 degree so that my Zigbee/Xbee shield can stacked onto the board.

As for those Components I needed are stated below:

LED: Lights up when ON/OFF and during triggered.



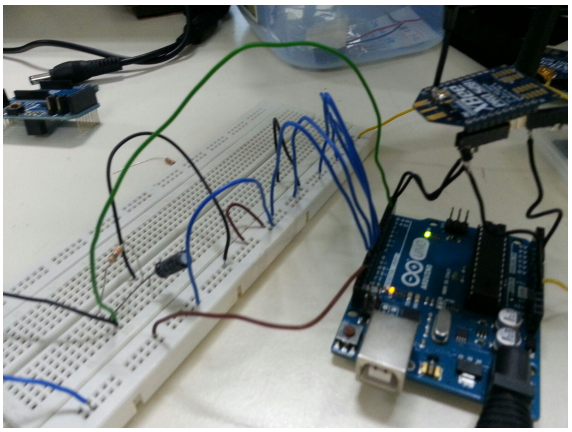
- Magnetic Reed Switch: Magnetic Sensor that comes in pair in which one of it will be connected in serial and both of it place it in parallel on the actual prototype. The magnetic reed switch will be connected in N.O.(Normally Open) meaning placing it together, the circuit is OPEN and when it is not place to each other the circuit is CLOSED. This will be the signal generated for triggered of mouse trap.

- Dip Switch: Allow end-users to set the TrapID of the traps. This switch will be binary representation of whole numbers and based on end-users preference, they can change the ID whenever they want.

- DC male power port: Connector between the Arduino Uno to the Battery Holder

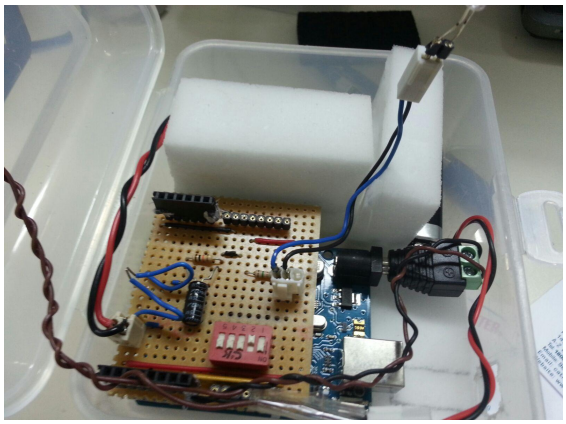
- Battery Holder: Power Source for the Arduino Uno

Those are basically the main component to make the slaves works and tested everything using the breadboard to make sure I did correctly before I transfer the circuit to my copper board.

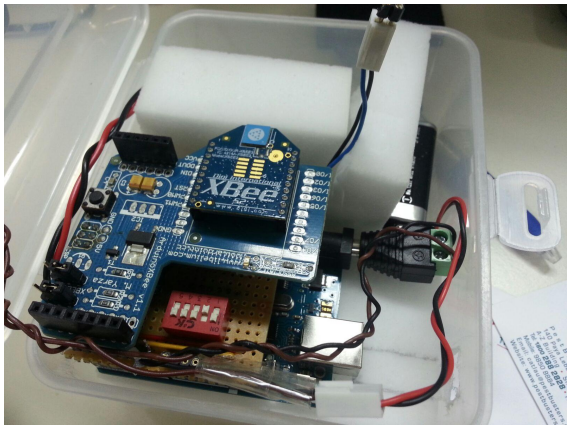


After the code is done and the the connection is correct, I transfer the components to a copper board and solder onto it. Copper board is about the same size in relatively to the Arduino board.

I also added headers to the side of the copper board so that copper board can be stacked onto the arduino board together with the Zigbee/Xbee shield.



Finally stacked the shield on top of the copper board.



First implementation of the Prototype

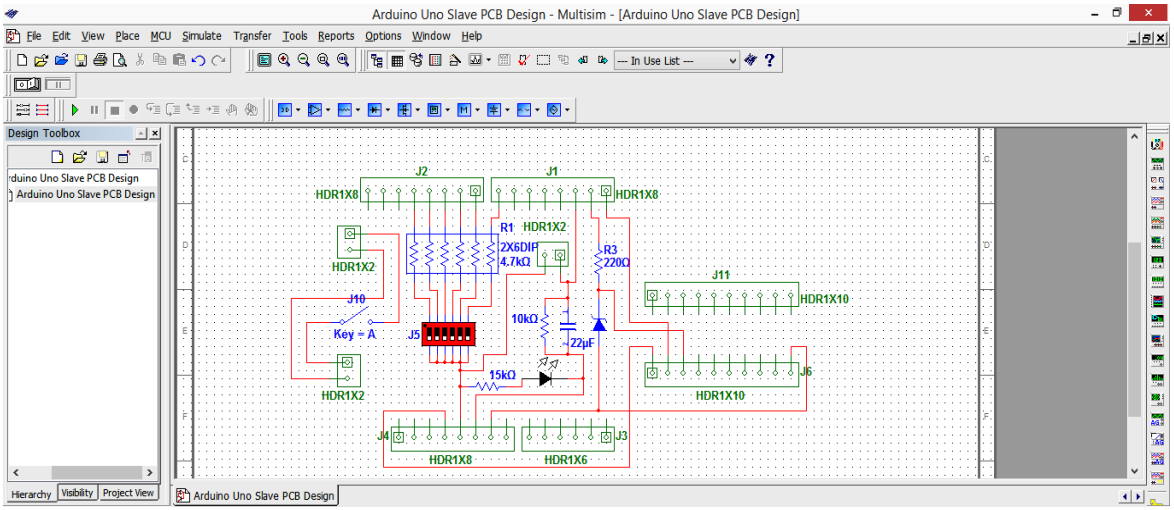
There is some problems to the power usage of the prototype, it will be ON throughout the whole time during monitoring and the power usage of it is 60mA. Based on using a power bank of 10000mAH, it will only last 166.67hours which is only 6.97≈ 7 days max before employee have to come down and change the power bank.

Even if I enable sleep mode on the Xbee, Power usage will drop to Avr. min. 30mA which will lasts ~14 days. Which is still far from the goal of 2 months.

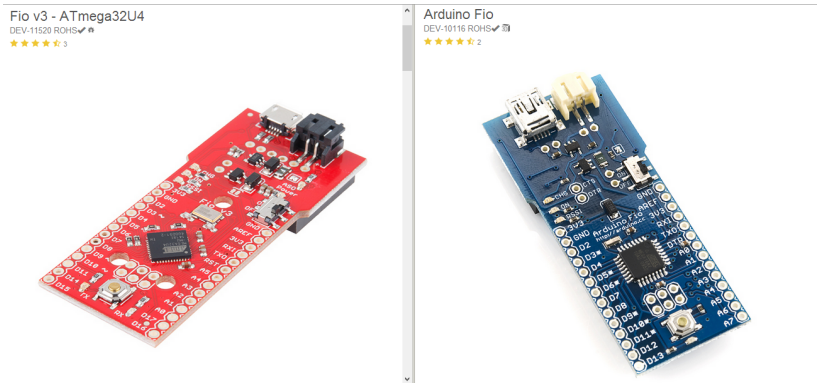
As in the picture, this implementation is already a failure as the copper wire are twisted around the plastic box and the shield has blocked the path for the copper wire to plug in and I have to directly plug in between the stacked to make it work.

Same thing, due to time constraint, We have to showcase the prototype to the industry.

After showcasing, I took the time making a PCB board to replace the copper board which it took quite some times to design and draw.



However all this changes as we are finding a solution for lowering the cost for the slaves. It seems that Arduino Fio v3 is able to replace the whole system for slaves as well and therefore, scrap the idea of doing it on Arduino Uno and headed forward to implement on the Arduino Fio.

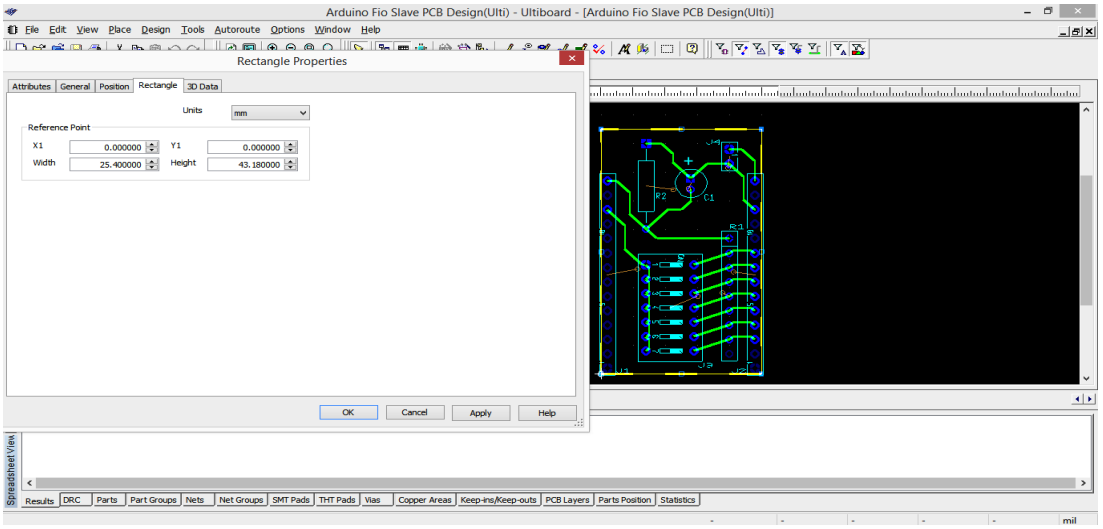


This Arduino Board, Arduino Fio is the solution to the cost of an Arduino Uno. It has a built-in Zigbee/Xbee slot which we can save up the cost of an Arduino Xbee/Zigbee shield.

Blue in colour, the Arduino Fio that is the old version from Arduino site which cannot compile and upload code onto the Board directly through the micro usb serial port.
Red in colour, the Arduino Fio v3 from sparkfun site does that function to solve the limitation of the old version. Therefore we went ahead to get Arduino Fio v3 and scrap the Arduino Uno Board.

It is a very big risk we are taking as a group because we only had a one time budget, once anything goes wrong, time and money will be wasted and the industry would not pay any more.

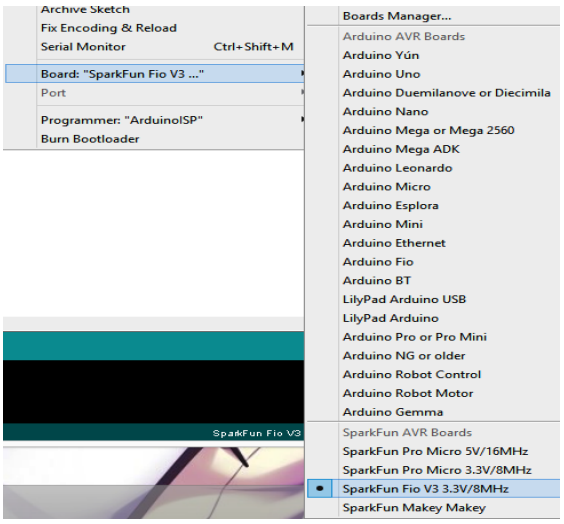
During the next week while waiting for the Arduino Fio to be in, I went ahead to design a new PCB board using the dimension given from the site.



The Size of the PCB board is 25.4mm to 43.18mm.
As I send in my design to my supervisor to outsource it for printing. They do not allow PCB board to be design as my board is less than the minimum condition to be able to print and wanted me to redo. But after reconsideration, I stick back to soldering as it takes lesser time and I will be able to use and reuse it straightaway.

On the same week, Arduino Fio V3 has arrived, but there is a big problem, when we start writing and compiling codes onto it, it kept showing errors. I found out later that Arduino Fio v3 is developed by sparkfun which the Arduino software does not support, therefore external package had to be downloaded and placed in the same library file as the other boards as well so we would be able to compile.

Therefore, I search for the port parameters of the Arduino Fio v3 that the board is operates on and gotten the parameters file for installation and updated a new drivers into both of our computers to work with. After doing some editing on the Arduino Uno code, everything works and ready to do hardware testing again.

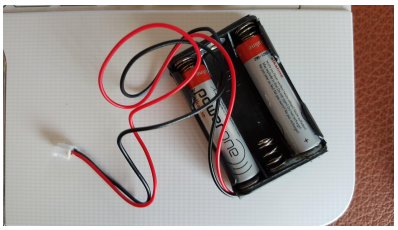


When I am going to transfer the components from the Arduino Uno, we heard feedbacks from the employees at pestbusters that they do not know how to set the trapID using binary so ended up we took out the Dip switch and code the trapID in software, DC power male port is also not in use as the Arduino Fio is powered through jst connector instead. Now the component left that will be reuse on the Arduino fio would be the LED and the Magnetic Reed Switch.

I added in:

Push Button: ON/OFF Button

Replace 3AAA battery holder with:

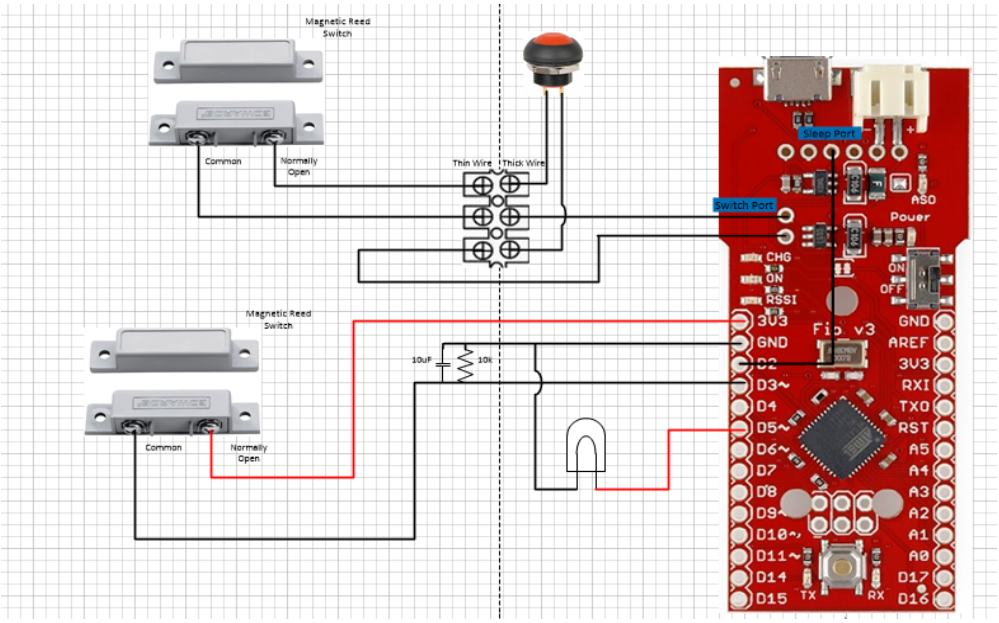


Battery holder with JST connector

To solve the power usage problem, I added a new pair of Magnetic Reed Switch to the system in parallel to the triggered sensor state. Instead of keeping the system ON and waiting for rats and send data to the Master when it triggered, I just turn make the system OFF when the trap is not triggered and turned the system ON when the trap is triggered, at the same time the sensor state will be read from and send the data to the Master.

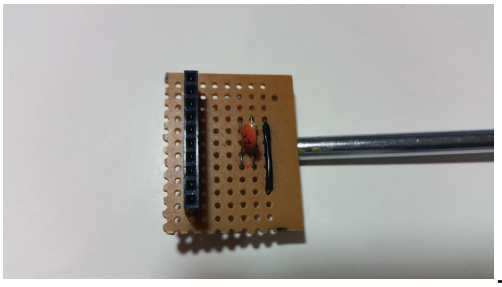
Therefore during that period of time when it is not triggered, the system would be able to save a lot of power.

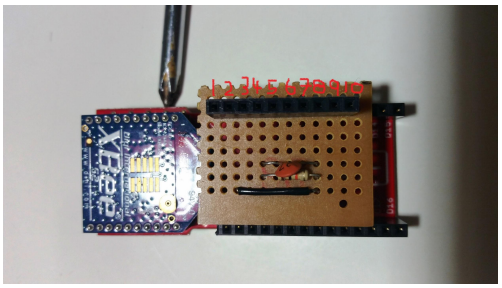
To allow the system to turn ON/OFF with the Magnetic Reed Switch, I added a connector block to it so that end-users is able to ON/OFF externally.



This is the final concept sketch of the slave

Solder board that replaced the PCB Board

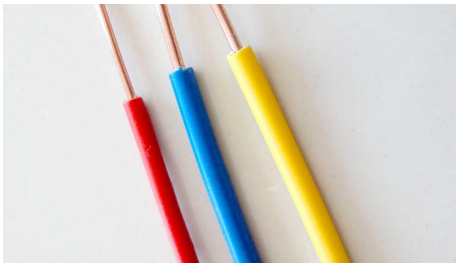


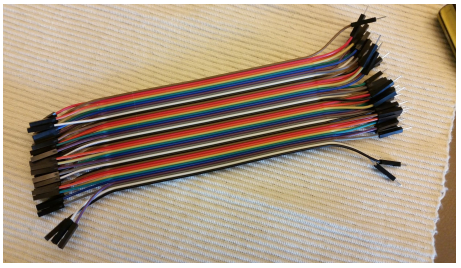


This piece of solder board will be stacked onto the Arduino Fio v3 and the cables will be plug on the headers above.
Signal that produce by the magnetic reed switch is always fluctuating and therefore, I (Jia Chen) included the capacitor so that it will output a surge of current into the input so the Arduino Board can read accurately, resistor added parallel to the capacitor serve 2 purposes

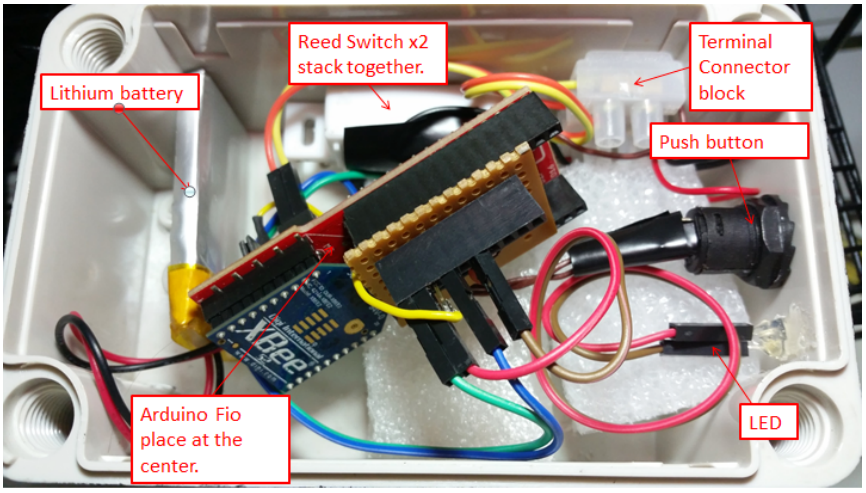
- To limit the current drawn, thus saving power consumption
- Protect my components from power surge damages

Instead of using the normal copper which will break easily, we acquire rainbow cable to plug onto the Arduino Fio v3 Board so the copper cable will not break in the headers.





Final Hardware Layout:

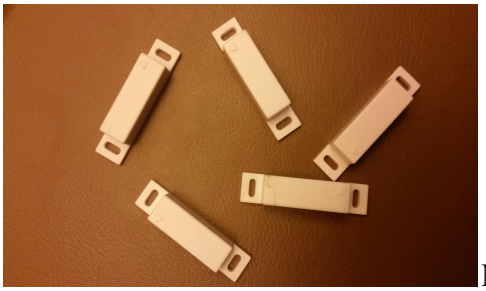


*Note that even Lithium battery with JST connector can be use as well.

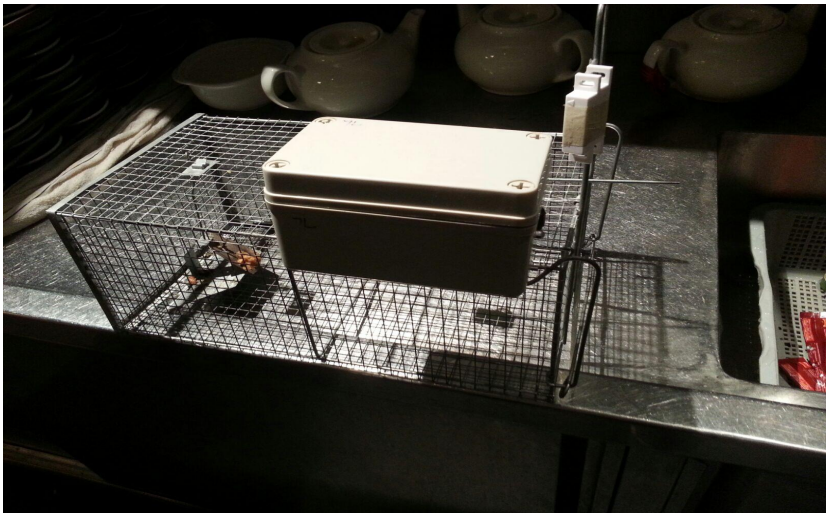
Since is an industrial project, they would want to test the prototype in real site and hence wanted a couple of sets.

Same as the Master, the casing itself is rigid and it is water-resistance, good for deployment on trial.

Each of the box will be mounted onto the spring trap and ready to deploy, we also mount one of the remaining magnet to the handle of spring trap so that when the trap door is being pulled back the magnet should be aligned with the magnetic reed switch inside the box.



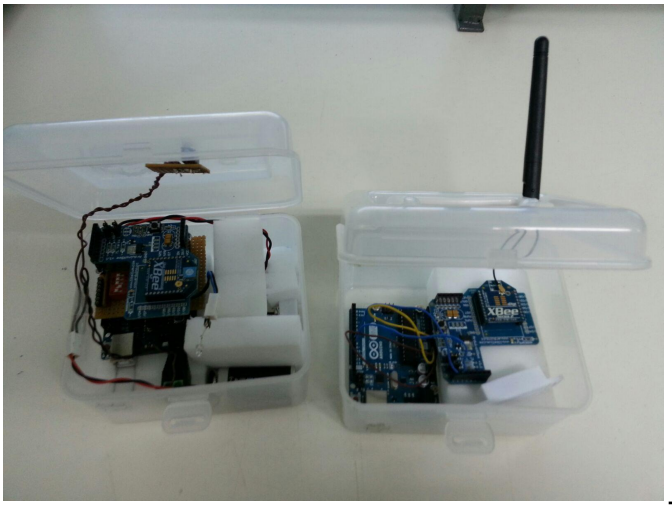
Magnets from the Magnetic Reed Switch._____



Final Prototype for slave

This is one of the trap that is ready to be place above the ceiling, rat bait is smudge on the trigger and ready to pulled back the spring and deploy.

From First Prototype to Final Prototype:



PROBLEMS ENCOUNTERED (HARDWARE)

Some of the problem I had already stated above and provided solutions for it, this parts is more on the tedious part of the hardware, which is also problematic for me as well.

Designing of hardware diagrams has been hard for me, I had to look at different schematic diagrams and the documentation of the Arduino Uno(Previously) and the Arduino Fio v3, even the documentation of xbee as well. Testing of range is difficult as there might be chances of loose hardware or the software code not working properly. Hence, troubleshoot for the traps is very difficult to pin point.

Coding Arduino software to set hardware parameters is the hardest as i need to know what is the register bits needed to set especially the setting for WatchDog Timer, i have to power the Arduino board, together with my component and measure the current using a multimeter and

see from start to end is there a current drop after sleep mode then after 8 seconds(code to interrupt sleep after 8 seconds) will it shoot right back up to the normal operating current.

Since i was doing it at the same time with the range test, it has wasted me a few hours just to see the correct result, before i let Ting Wei to use that function in the loop().

Soldering of Female headers on the Arduino Fio takes times as well as the soldering of the copper board. Before Rainbow cable was use, the single-core copper wire also have to solder on to the header pin and since is so small it took me 5 min just to solder one pin total there is 5 pin.

Slaves casing is also causing a problem as the casing itself is quite tight, some metal part are exposed and i am afraid that some component may touch until the power line and spoilt the whole set, i took the time to use insulate tape to cover up each of those exposed metal part.

All in all, the process is dreadful and it is an ongoing process.

2.3 Software

For the software development, we had used a few programs for our project like Arduino, Android Studio, Access, MySQL and phpMyAdmin. Arduino is used to communicate with the Zigbees between the slaves and the phone. Android Studio is used for creating applications for android. The applications are programmed to communicate with the slaves and the database. Access is a tool to create local databases and can be imported to MySQL. MySQL is a software where it manages the database of the customer details, mouse trap locations, as well as the logging of the mouse traps. phpMyAdmin is used to work along with MySQL to do administration.

2.3.1 Arduino

Arduino Code consists of setup() and 6 main functions to be called by loop():

checkAck(), battstat(), replyMaster(), getSencsorState(), sleepNow(), watchdogEnable().

loop() will be executing the whole arduino board which I will explain it in flowchart how the whole process works.

Functions: checkAck(), battstat(), replyMaster() - By Ting Wei
These functions allow data to communicate between the phone and Arduino board which includes sending of replies and acknowledgement. They also include how long to each of the data to send out without data collision (delay).

Functions: getSensorState(), sleepNow(), watchdogEnable() - By Jia Chen
Allows hardware to send signal to the Arduino Fio v3
Which includes 2 interrupts, Signal interrupt from falling edge of the Magnetic Reed Switch and Watchdog Timer interrupt whereby the whole system shutdown and only wakes after the timer is up.

Each of them will be given a description of what it does:

setup(){

Use to initialise the Arduino board to a specific baud rate for the code to compile, the trapID (eg.001, 002) and the pin whether is an INPUT or OUTPUT. trapID having 3 digit is to be able to set up traps >100 in actual future deployment.

Pin are set as input for reed switch because that will be the pin that the arduino board will read the state from (0 or 1) and output to an LED and also output to Xbee so that it will go to sleep after the whole operation. There is also an "ack = "ACK"+trapID;" that it will be compared to checkACK() function which I will be explain below.

}

checkACK(){

This is where the Arduino board would get its data from the Xbee Rx which makes sure that the master has received the triggered information. When the Arduino board has the acknowledgement from the master which is a string that contains "ACK" and the ID of the trap (e.g. ACK005) , this is when the comparison takes place between the received string and the string from setup().

Once both strings match, it will call the function sleepNow() to allow the Xbee to disable its transmitting ability and go to sleep. Hence, this will save up the power consumption and makes the battery last longer. We do this comparison is to confirm that the data it sends out from the Arduino Board is the correct trap triggered so other traps will ignore since the data is being broadcasted to other traps as well.

The trapID is the one that differentiates one another so if the end-users set the same trapID for 2 traps, depending which trap is triggered first, the latter one will not work.

}

battstat(){

This is used to determine the estimated battery life that is being plugged into the Arduino board. It is able to read the analog of the board where we used a voltage meter to find the highest point and the lowest point of voltage values it can operate. We decided to take the values' difference and graph it in percentage form.

However as every battery is different, the highest and lowest points of voltage may vary. Nevertheless, it allows the end-users to know the battery is low and must be changed.

This battery status information is sent to master together with the trap information when trap is triggered. However, it is for the application to decide whether it wants to display battery status or not. Meaning in testing mode, the application will convert to user-friendly status with the battery information. As for the monitoring mode, it will change automatically to for example “Mousetrap 001 is activated” the moment it gets the ID.

}

replyMaster(){

This function is used to reply any data back to the master, whether it is triggered information or acknowledgement information. It is able to display the messages on console as well to do troubleshooting to see if there is any malformed, cut-off or garbled strings being sent to master.

}

getsensorState(){

This is where Arduino Board will read the sensor state of the Magnetic Reed Switch from the INPUT pin that we initialise in setup(). When the trap is deploy on out field, the trap state is under logic ‘0’, once the trap is triggered, Arduino Board will read it as ‘1’ and this will allow loop() to send triggered information to its master.

}

sleepNow(){

Able to set the different sleep mode on the Arduino board, capable of shutting down from just clock cpu to shutting down the whole active clock domain in the board. 5 modes is available for use: Idle (least power saving), ADC Noise Reduction, Standby, Power-save and Power-down(Most power saving).

In this function, There is the attachInterrupt(0, sensorState, FALLING); single line of code, which means it will interrupt the arduino board from doing other activity when the signal of the Magnetic Reed Switch reach the falling edge, which will disable the interrupt from sensorState() function and jump down back to the loop();

There is also need to set additional functions to make it worked: sleep_cpu(), sleep_enable(), sleep_mode(), sleep_disable(). All this will:

1. Enable the sleep bit in the Arduino Board register.

2. Really let the Arduino Sleep when the loop() code reaches here.
3. What sleep mode to be in.
4. Continue where the code be when it is being interrupted.

As our setting for sleep mode is SLEEP_MODE_POWER_DOWN, only watchdog timer can wake the Arduino and this will interrupt will be implemented in watchdog() function.

By right, under normal operation, the Arduino Board will straightaway go to sleep and not wake up but there is some issues or wrong operation might happened which I would explain in watchdogEnable() function call.

}

watchdogEnable(){

This is the interrupt call function, what to do here is to set the register bits in the Arduino board to use the built-in Watchdog Timer

Well in the loop() when watchdogEnable() is called, xbeeSleep pin will be set to 1 as well. Therefore when the Watchdog Timer starts, Xbee goes to sleep as well.

To wake the Arduino Board up, ISR(WDT_vect) sub-routing has been call in the loop(). what it did was to disable watchdog timer after time is up.

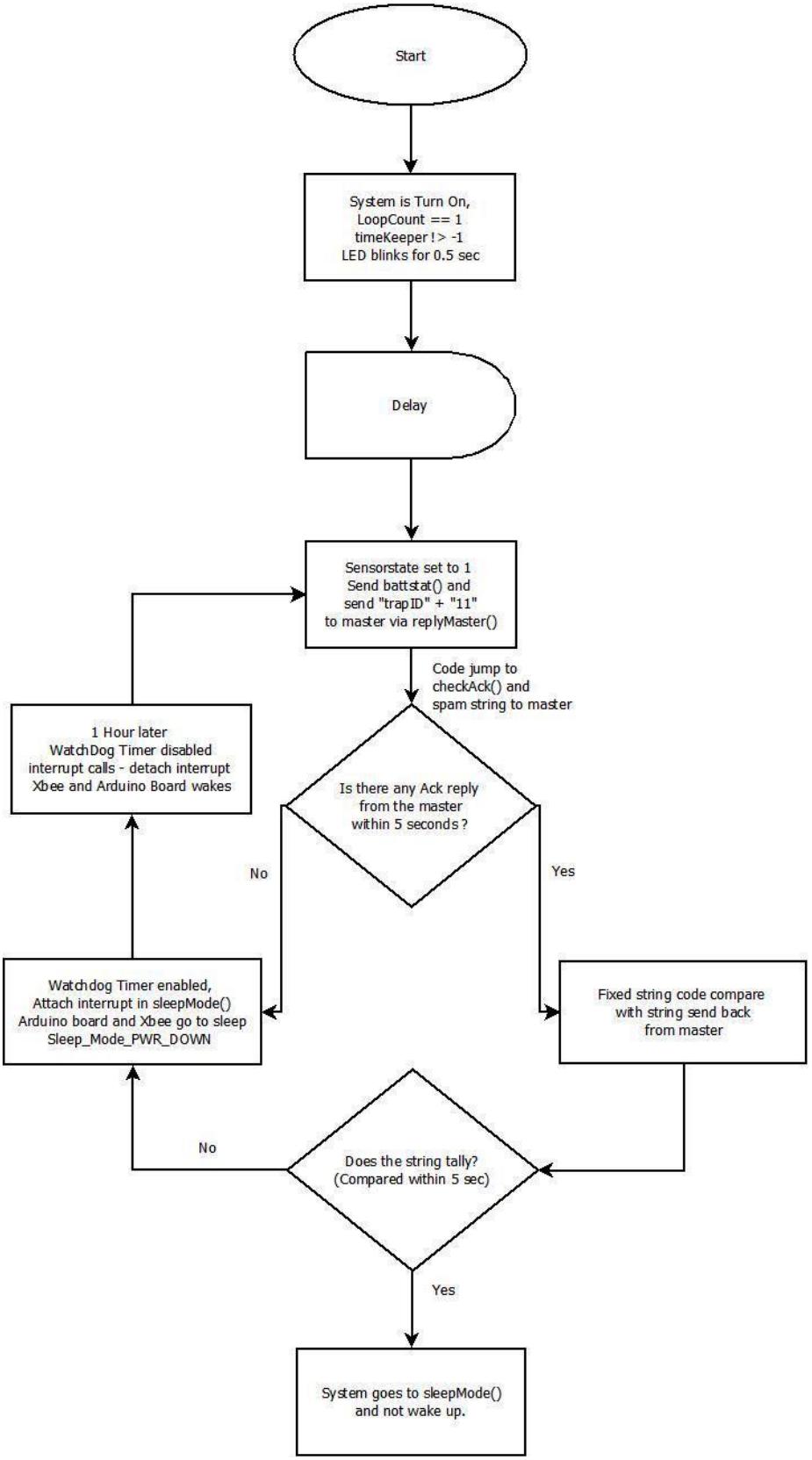
However watchdog timer can only be set up to 8 seconds in the loop() and will interrupted the board to wake up. Hence, in the loop(), what was did was to repeat the 8 seconds timer for over an hour so it will wake up every one hour to send the triggered information.

This one hour will start if the trap which had activated send the info over to master but did not send back an acknowledge info to the trap at start and therefore, to get the acknowledge info from the master, the system will go to sleep first. After one hour, interrupt will be called to handle this issue.

}

loop(){

Flowchart of Arduino Code:



}

2.3.2 Android Studio

Android Studio is an integrated development environment (IDE) and it is using mainly Java programming to develop applications specifically for Android. It is suitable for our project as the applications are available for users who already had an Android phone. In this project, we have created two applications and they are RatzManifest and RatzManifest Monitor.

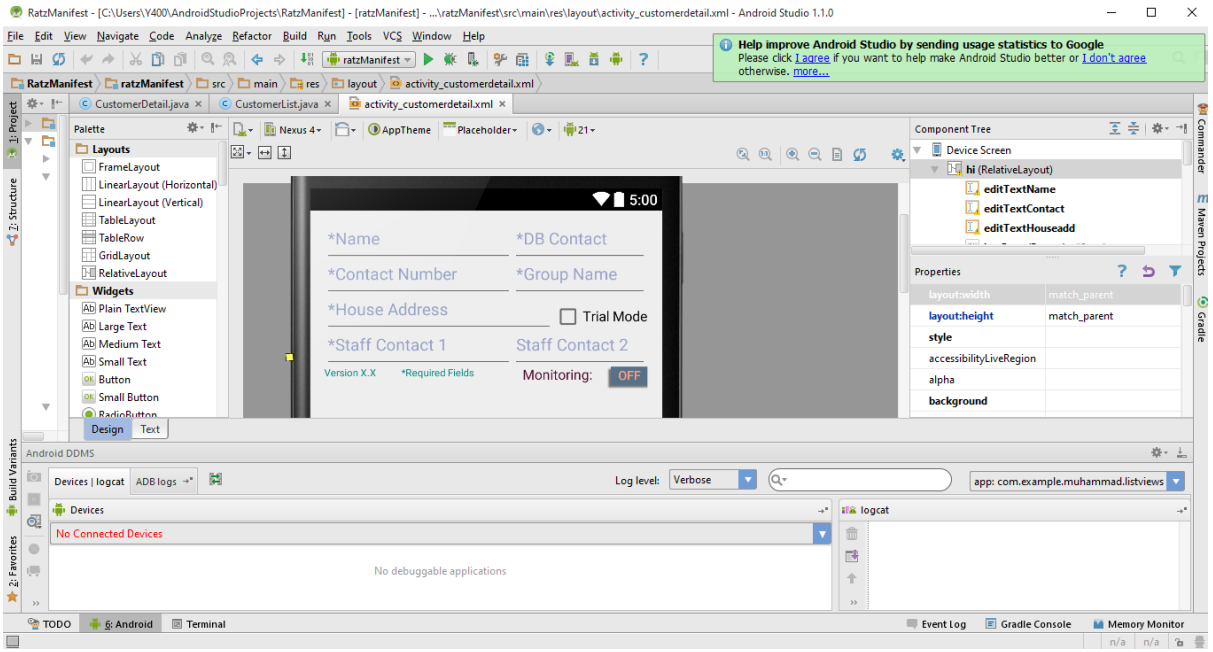


Fig. 1 Android Studio Interface

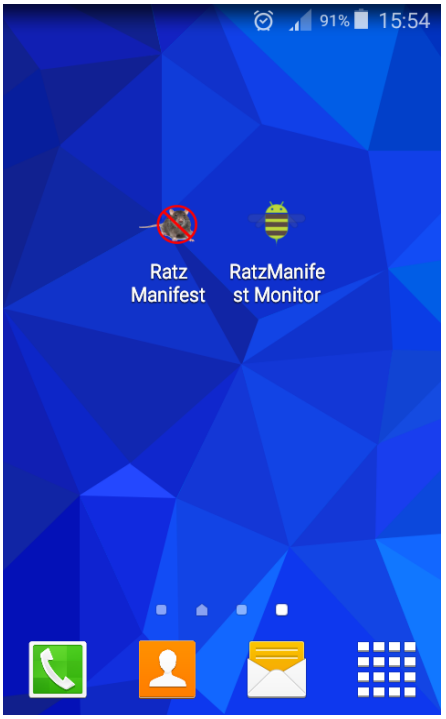


Fig. 2 Two applications: RatzManifest and RatzManifest Monitor

RatzManifest is the main application that saves and manages customers' information, monitors the mouse traps and sends SMS messages to the staff and the external database. The content below shows how the application is made.

Splashscreen activity

The main purpose of a splash screen is to display an image for a while before the application moves on to the CustomerList activity. The image can contain of the name of the application, logo and the current version.

Fig. 3 Splash Screen

CustomerList activity

This activity will appear first after displaying the splash screen. This contains a list of names of the customers. When the user taps on one of the names, the application will create a CustomerDetail activity. Usually when the user first opens the application after installing, the list will be empty and the user will be greeted with "No Customers!". To add a customer to the list, the user has to create a new customer by tapping "Add New".

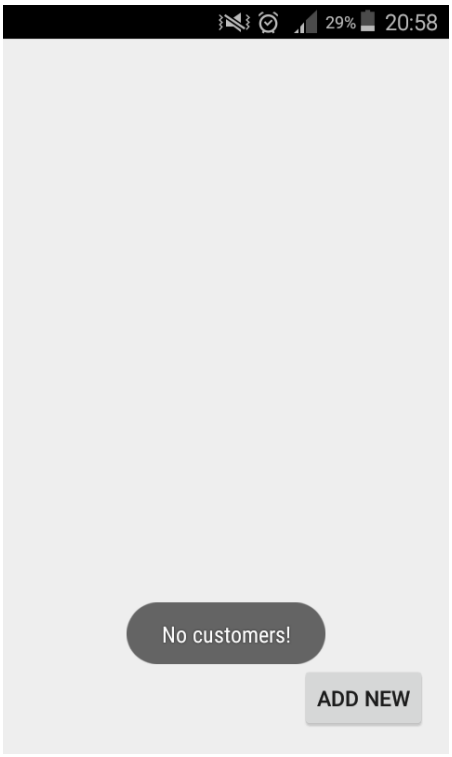


Fig. 4 Empty CustomerList layout

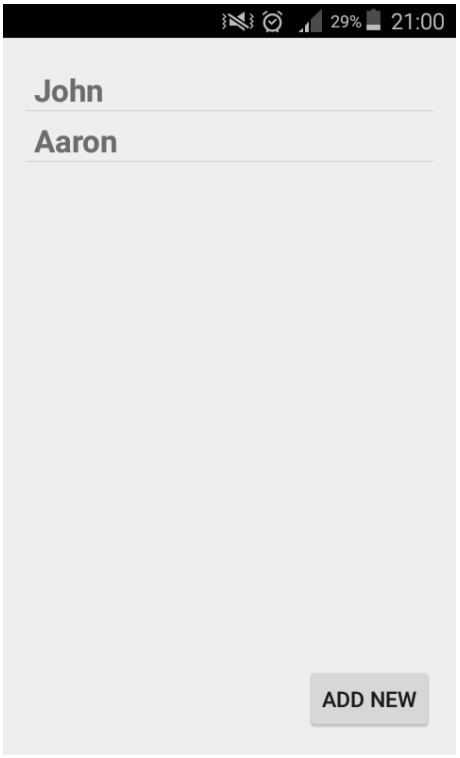


Fig. 5 CustomerList layout with customers

CustomerDetail activity

This activity is shown when either the user wants to create a new customer or selecting any one of the customer names. The activity consists of textboxes (e.g. Name, DB Contact, Contact Number, Group Name, House Address, Staff Contact 1, Staff Contact 2 and MouseTrap Locations), buttons (e.g. Start, Test, Delete and Save), an options menu (e.g. Preferences, Stop Monitoring and SMS Database Options), a spinner (e.g M001-M127), a checkbox (e.g. Trial Mode), a listview and lastly labels/textviews.

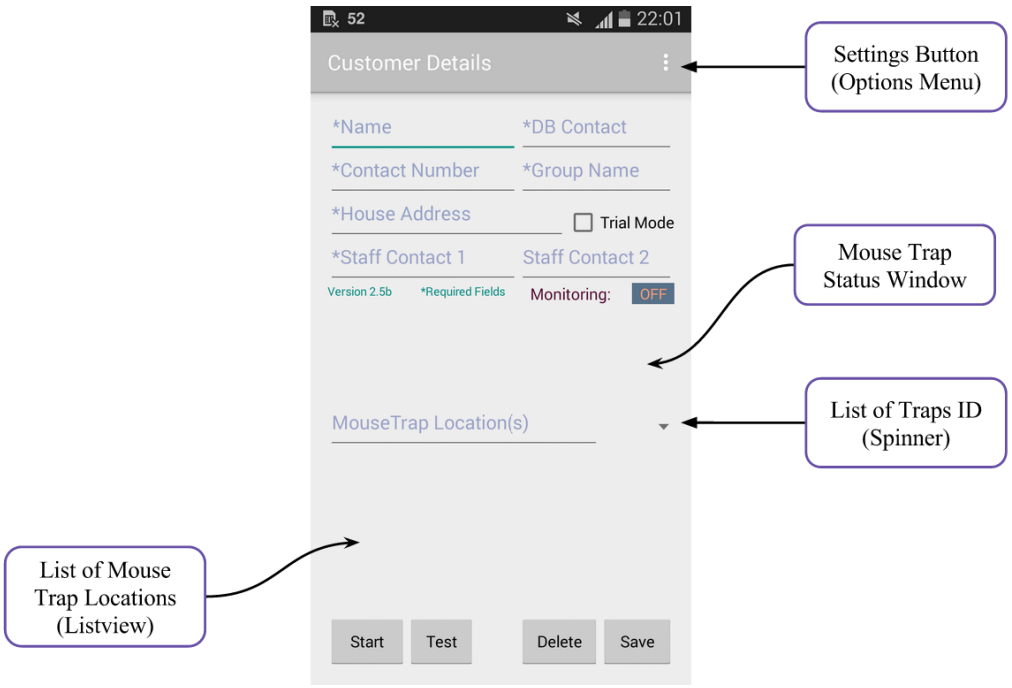


Fig. 6 Overview of CustomerDetail layout

The user will have to fill in the textboxes with the customer details. DB Contact textbox is used to send SMS to database. Staff Contact textboxes are used to SMS up to 2 people to notify them about the traps being triggered. The mousetrap status window is to monitor the traps by displaying their status whenever they send their information. To add a new mouse trap location, the user has to write it into the MouseTrap Locations textbox. This is followed by selecting one of the mouse trap IDs in the spinner located at the right.

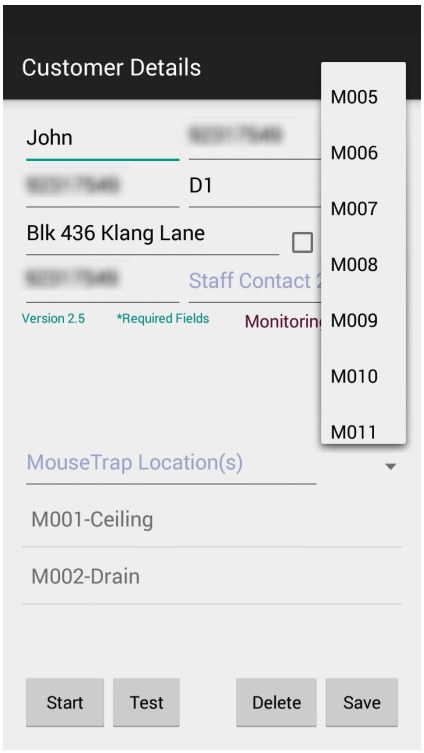


Fig. 7 Spinner showing traps ID

Moving on to the buttons, the Save button is used to save customer's information to the local database, and then SMS the save information to the external database. The sending of SMS messages to the external database depends on the length of information of the customer's details, as well as the traps ID and their locations. As SMS has a character limit, a special algorithm will be used to split the information into multiple SMS messages before sending if the length of information exceeds the limit of one SMS. Delete button is used to remove the current customer from the local database, and then SMS the delete information to the external database to remove the customer from there too.

For the Test button, it is used to check the communication with the slaves whether they are responding. When any of the traps sends its status to the phone, the application will read the information and convert it into user-friendly. The information will then be printed in the window. The Start button is mainly used for monitoring the traps. This will enable the application to be monitoring mode, turning the monitoring label from OFF to ON. This leads to instead of converting when getting the status of the traps, it will SMS the staff and the external database.

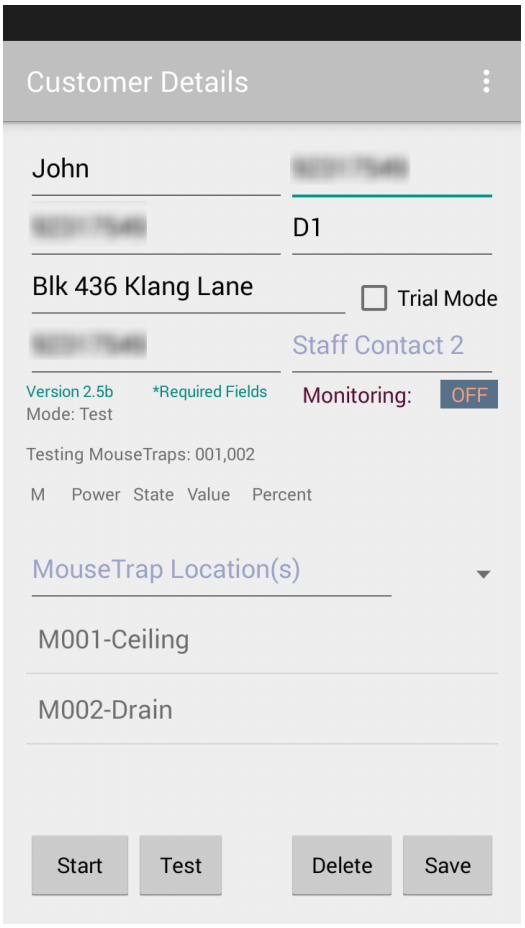


Fig. 8 Testing Mode

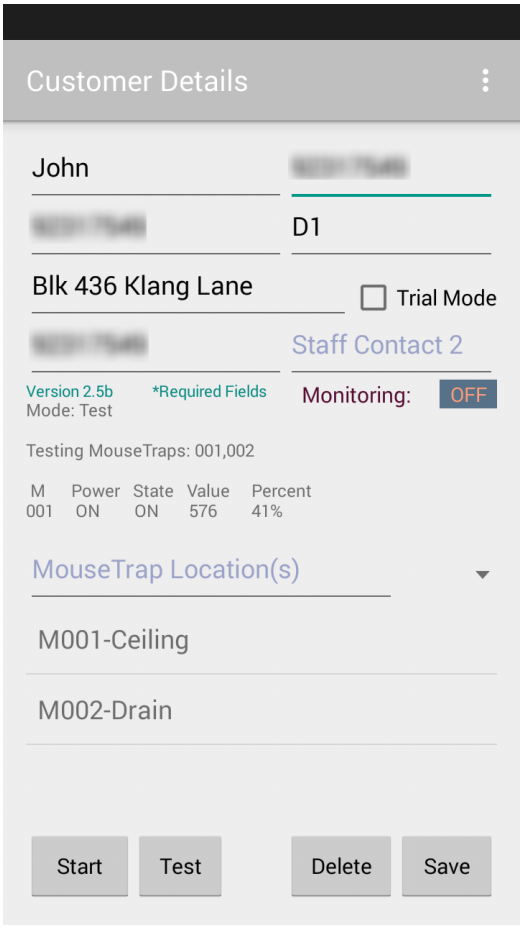


Fig. 9 Mouse trap status

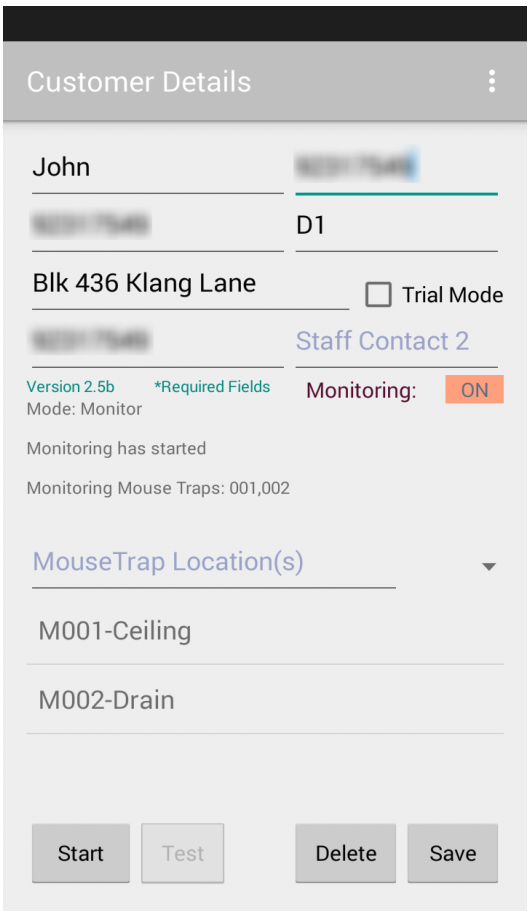


Fig. 10 Monitoring Mode

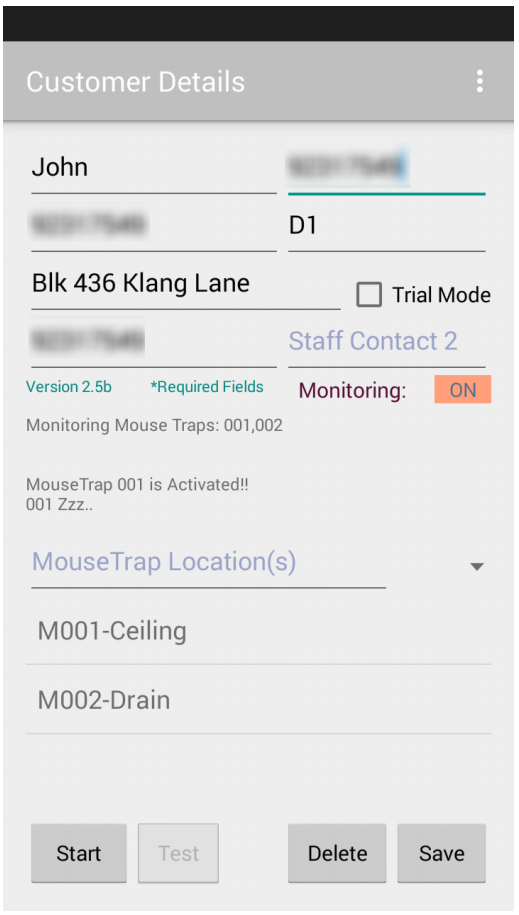


Fig. 11 Mouse trap triggered

In the options menu, the Preferences option is to allow the user to select either 31, 63 or 127 mouse traps if the user needs more trap locations. Stop Monitoring option is only used when the application is in monitoring mode, the user wants to stop the app from retrieving any more information from the slaves. SMS Database Options option is used to either enable or disable SMS to the external database. This is a safety mechanism as, for example if there are many mouse trap locations (60 traps) and the user accidentally selects Save, there will be multiple (around 10) SMS messages due to the character limit. Only when the user finalizes the information, he/she will then set the SMS Database Options to enable before selecting Save.

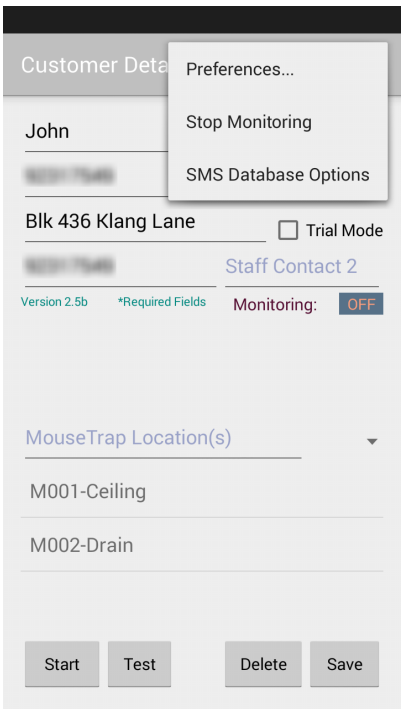


Fig. 12 Options Menu

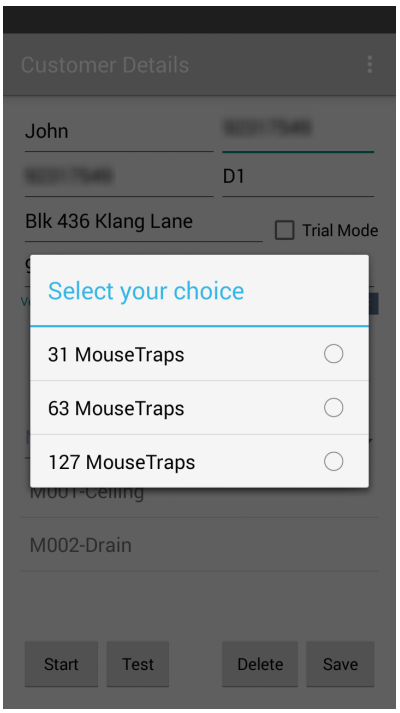


Fig. 13 Preferences Menu

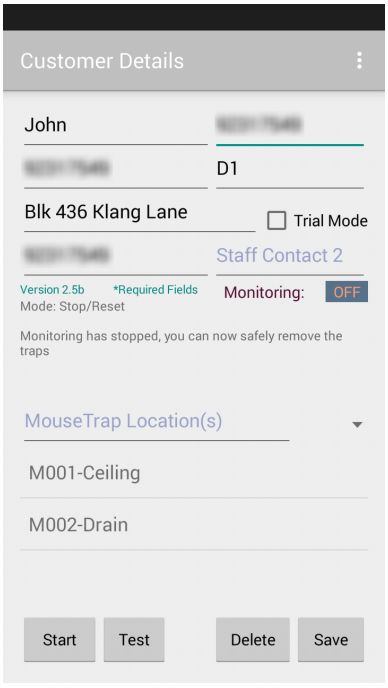


Fig. 14 Monitoring stopped

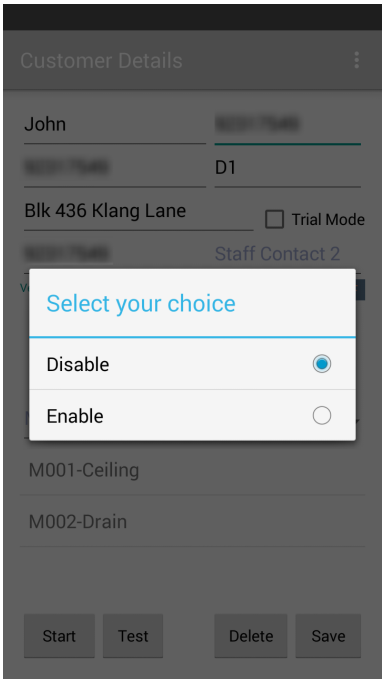


Fig. 15 Enable/disable SMS to ext. database

DBHelper.java

To save the customer's information and the mouse trap locations, a local database is required. In this application, SQLite will be used to store data into local database. This is useful as the user do not have to fill in the blanks again the next time he/she uses the application.

Flowchart of RatzManifest application:

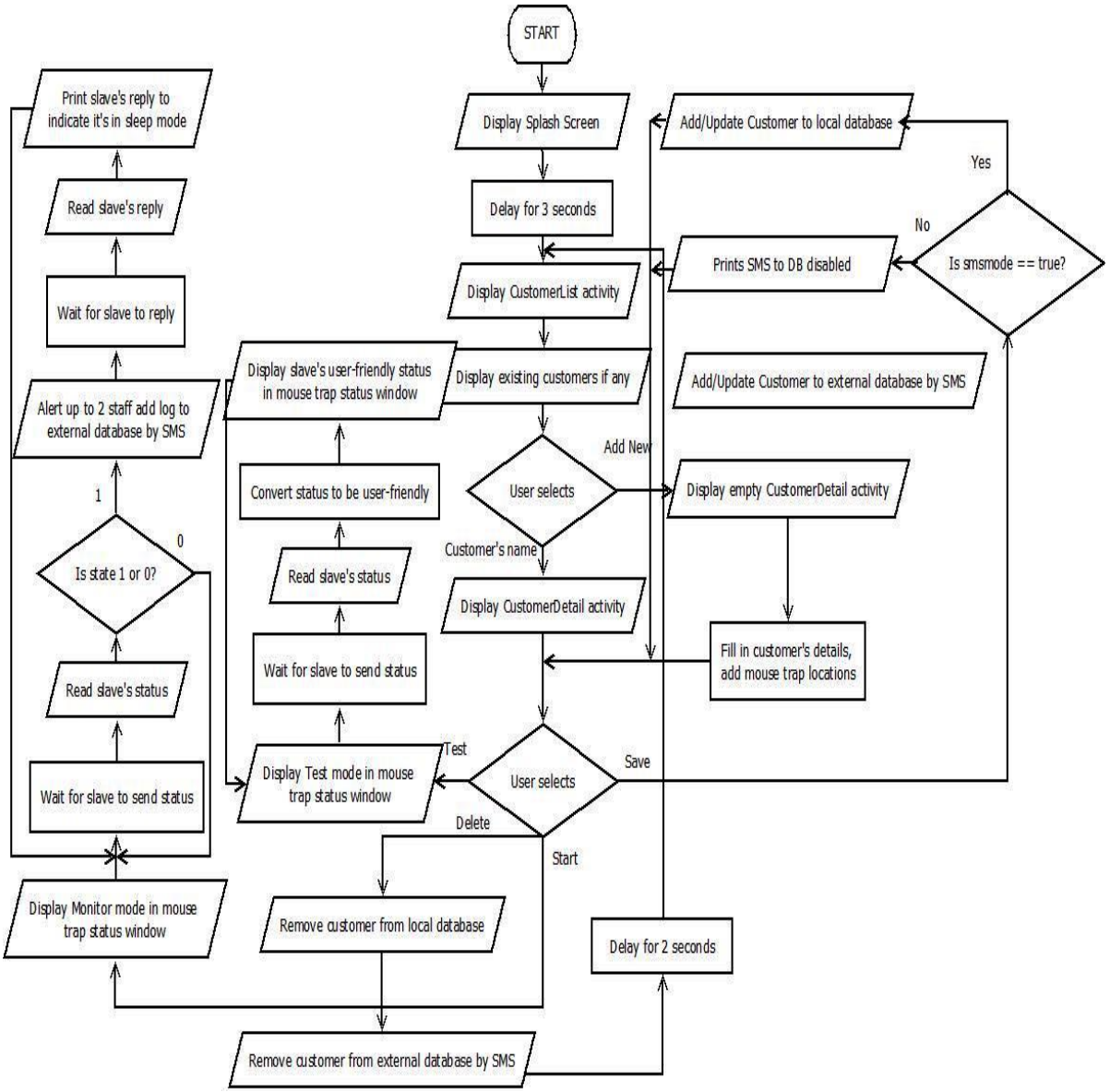


Fig. 16

RatzManifest Monitor

RatzManifest Monitor is a second application which did not exist at first as we were planning to finish and stop at the main application. Mr Afendi tried our prototype for quite some time, and thus he had some suggestions for us to consider. One of the suggestions is that he wanted to check the monitoring status when he was away via his phone. This means that an external database server is needed to store the information and can be viewed by anyone remotely via an application. This results in creating a second application as an improvement for the prototype. It is also to prevent user confusion and complication of adding more functions into a single application.

LoginActivity activity

The activity is shown first when the user opens the application. It is a login page where users can login to access customers' details and status of mousetraps. It consists of two textboxes (e.g. Username and Password), an options menu (e.g. Run in Background and Disable Receiving SMS) and lastly a button (e.g. Login).

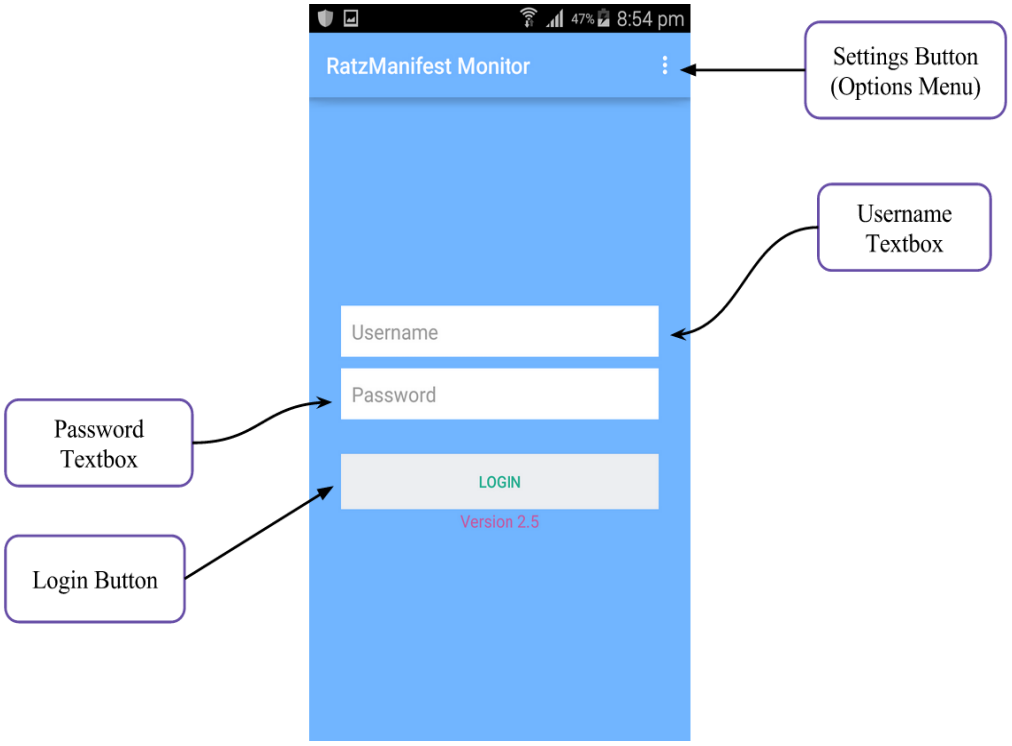
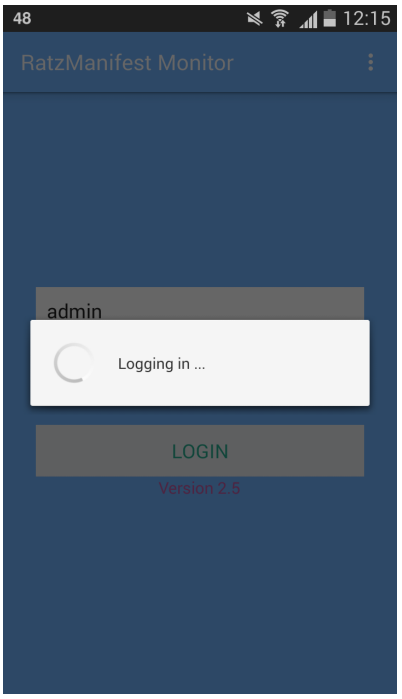


Fig. 17 Overiview of LoginActivity layout

To login into the main activity, the user will fill up Username and Password textboxes with his/her account details, then select Login button. If the username or password is wrong, it will display an error message: "Incorrect Username or Password!"



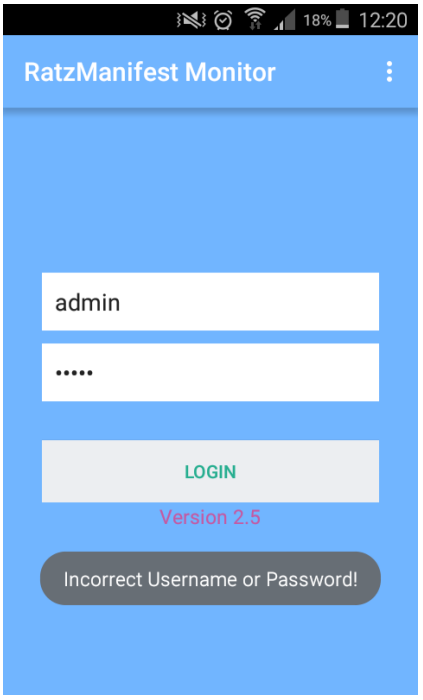
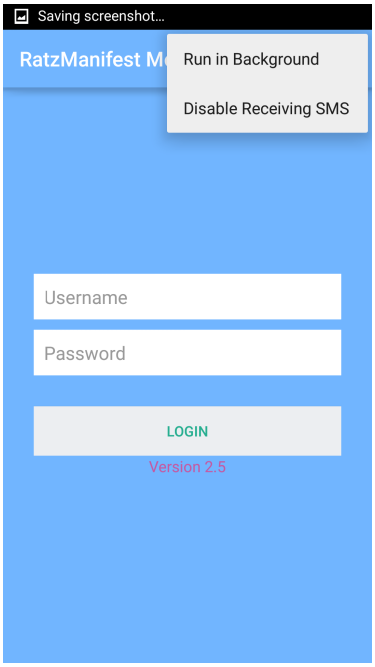


Fig. 18 Attempts to log in Fig. 19 Log in failed

The options menu has two options: Run in Background and Disable Receiving SMS. Run in Background allows the application to run as a service. This means that when the phone receives its SMS, the application will respond to it even though the application is not currently in use. Disable Receiving SMS option prevents the application to respond to any incoming SMS messages.



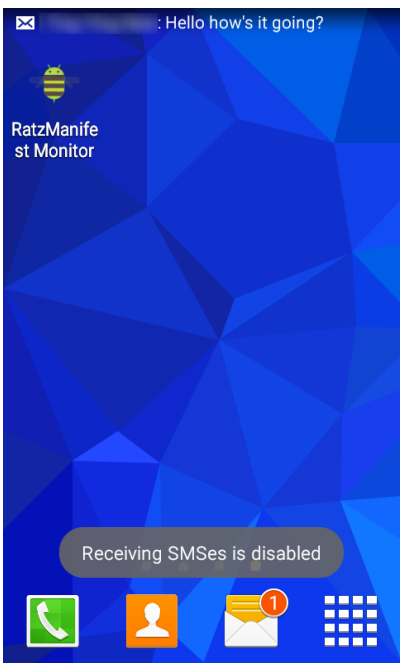


Fig. 20 Options Menu Fig. 21 Background Mode Fig. 22 Disables Receiving SMS

SmsListener.java

This java file is used to allow the application to listen to incoming SMS messages and the application will be opened after a specific delay to update the database server using that SMS. SmsListener will be called first whenever that happens. Depending on the user's preferences (e.g. Disable Receiving SMS), it will either store SMS messages into the local database or ignore them. Multiple SMS messages can be stored in the local database.

When does the data from the local database being used to update the external database? It is when the application opens, a special function is called in the LoginActivity and it will examine the data whether it is an invalid SMS or not. Once it is done examining the content, it will either update the external database if it's valid, or drop immediately from the local database if it's invalid. The local database will be emptied eventually when the login activity is in foreground. After it is done updating/dropping the data, it goes back to background mode automatically and waits for more incoming SMS messages.

MainActivity activity

This is where the users check customers' information, trap locations and the logging traps information. This activity consists of an options menu (e.g. Details, Log File and Traps Information), a listview and lastly a button (e.g. Logout)

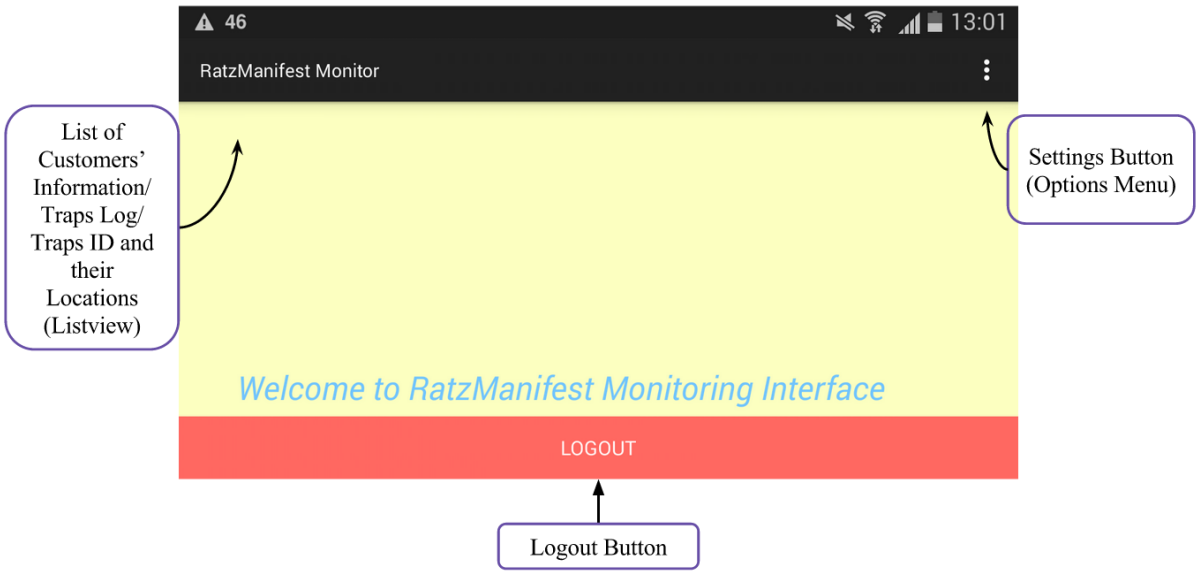


Fig. 23 Overview of MainActivity layout

Please note that for the next three options in the options menu, the application does not extract information from its local database. It queries from the external database and then displays in the listview. This will be explained in detail in a later section. For the Details

option in the options menu, it will display the customer's information. It shows the customer's name, his handphone number, his address, number of traps, first and second staff's handphone numbers.

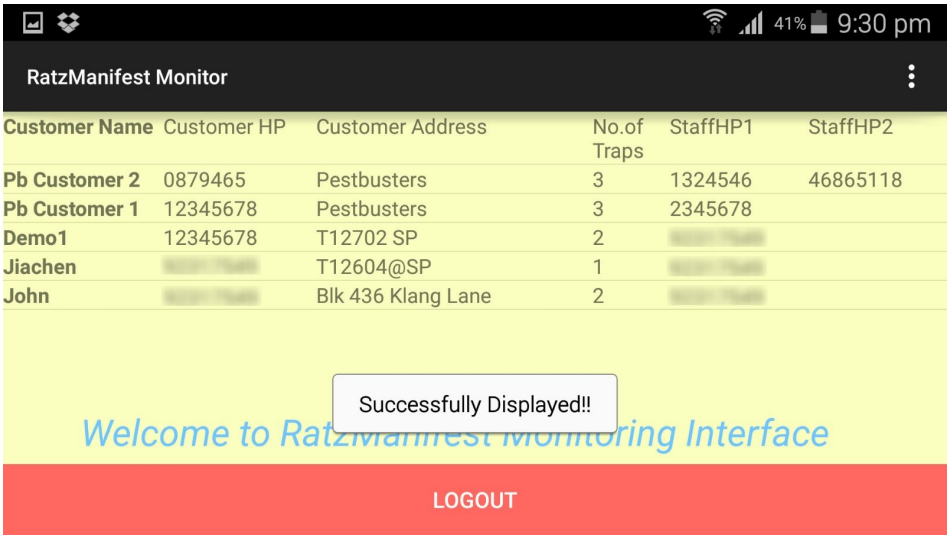


Fig. 24 Customer details

Next, the Log File option displays the logging status of the mouse traps It includes the customer's name, ID of the log, timestamp of mouse trap triggered, ID of trap and the trial mode.

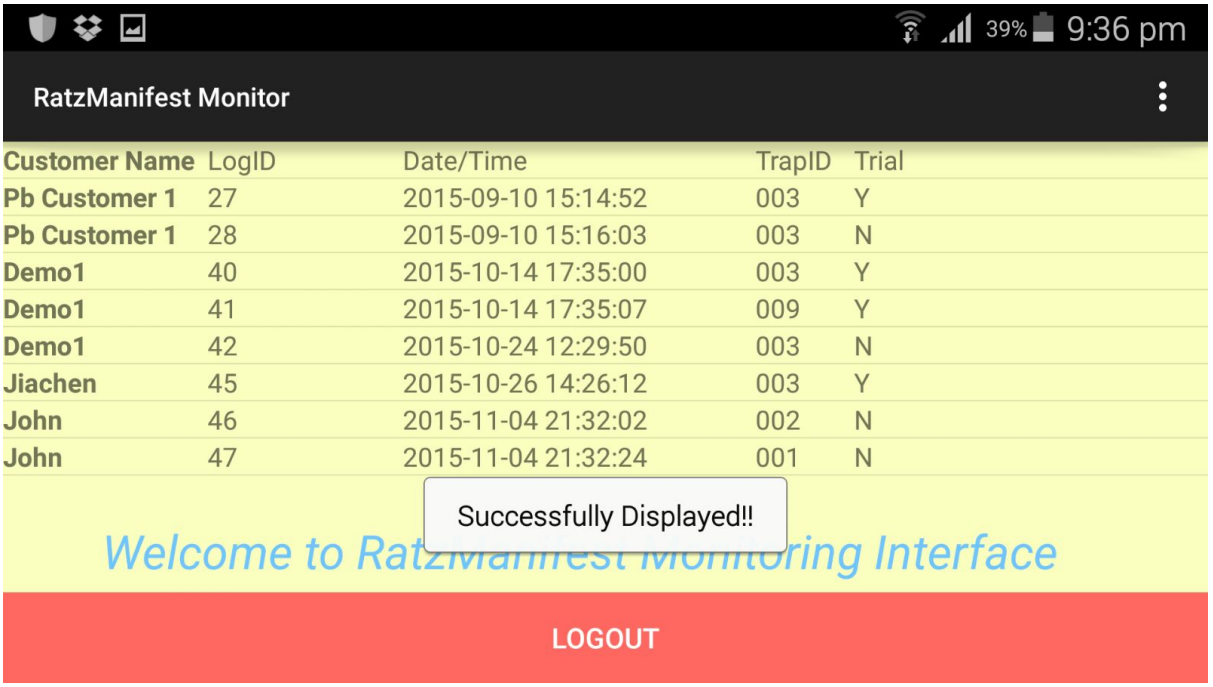


Fig. 25 Log file with timestamps

Last but not least, the Traps Information displays the customer's name, ID of trap and the trap's location.

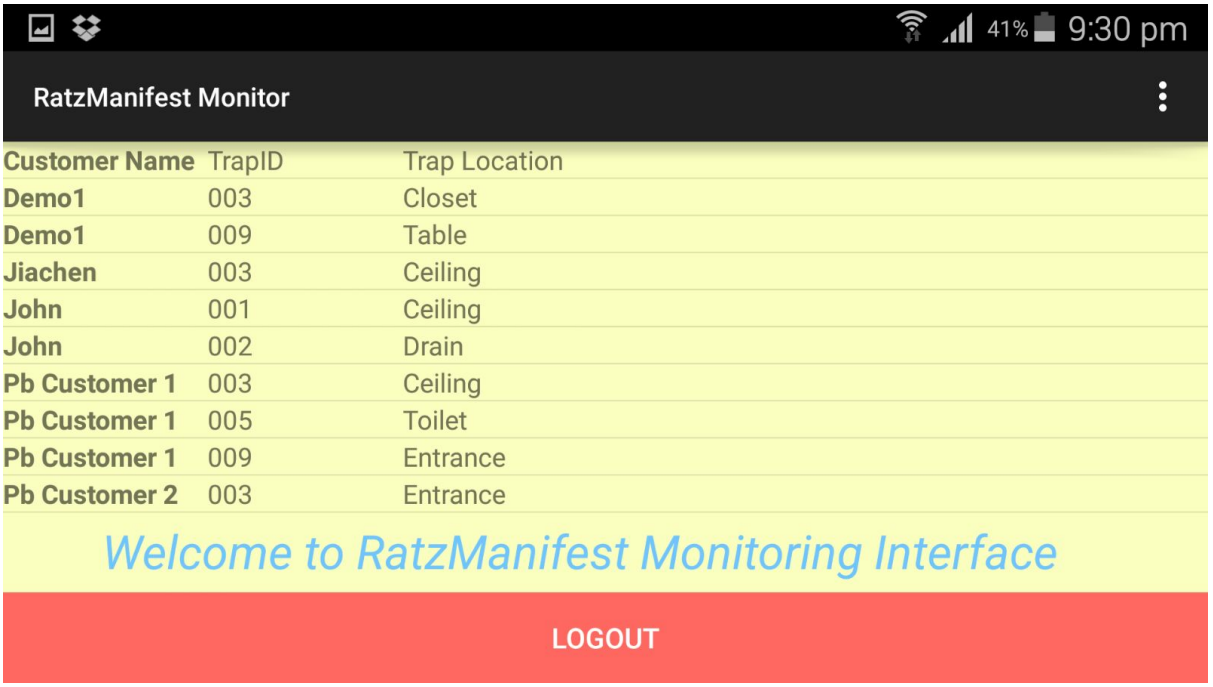


Fig. 26 Traps ID and their locations

PHP and SQL queries

There has to be some way to communicate the application and the external database server. Using StringRequest and PHP allow the app to connect to the database, and SQL queries are used to update the database. For example when adding new customer, his/her details are transferred to StringRequest. In the index.php, it receives the content and it will update the external database based on the tags. If the tag matches with any one of the tags written in index.php (e.g 'customer'), it will move on to do the respective SQL queries to add a new customer in the database. If there is no error, the PHP responds to the application, saying that it is successfully added. The php files are stored in the FTP server so that when making changes to the external database, the application will send information to where these files are located via a link.

2.3.3 Access, MySQL and phpMyAdmin

These two programs, MySQL and phpMyAdmin, are required to create an external database for the users to access it via the second application. Before creating a database for MySQL, we need to plan out the tables and their relationships. To make the tables clear and simple, we will use Access. In Access, there are four tables, Login table, Customers table, Traps table and LogFile table.

Since there will be many staff serving different customers, a login system will be required to allow authorized users to access. For example, there are two staff that serves 6 customers.

One staff is serving customers A, B and C while the other staff serves customers D, E and F. When the first staff logs in to the Main Activity to view his customers' details, he is supposed to see customers A, B and C information only. To achieve that, the administrator will give every account a unique group name. The group name in Login table will be checked with the group names in Customers table, Traps table, and LogFile Table. In this way, only information with the same group name will be displayed based on the account, and only the administrator account has the access to see all customers' information.

If there are no customer details, there is no point creating this application. Therefore, the Customers table is required to store all customer's information to allow the user to see remotely. The Customers table consists of customer's name, customer's handphone number, customer's address, number of traps, first and second staff's handphone numbers, and group name. Group name will not be displayed in the application as its main purpose is to filter out the customers. Customer's name is important to the other tables because there may be same trap ID or same trap location, thus this is to prevent confusion.

For the Traps table, it is stored into four fields: customer's name, trap ID, trap location and group name. Like the Customers table, group name will not be displayed in the application. Moving on to the LogFile table, it consists of log ID, trap ID, customer's name, date/time, group name, and trial mode. The log ID is unique and will increment by 1 whenever a new log is added. This is usually occurs when the trap is captured, the customer's name, trap ID and group name will be recorded. The date/time is taken from the current time of the database server when a new log is added. The trial mode is to indicate whether the log is from the trap triggered by simulation or by live monitoring.

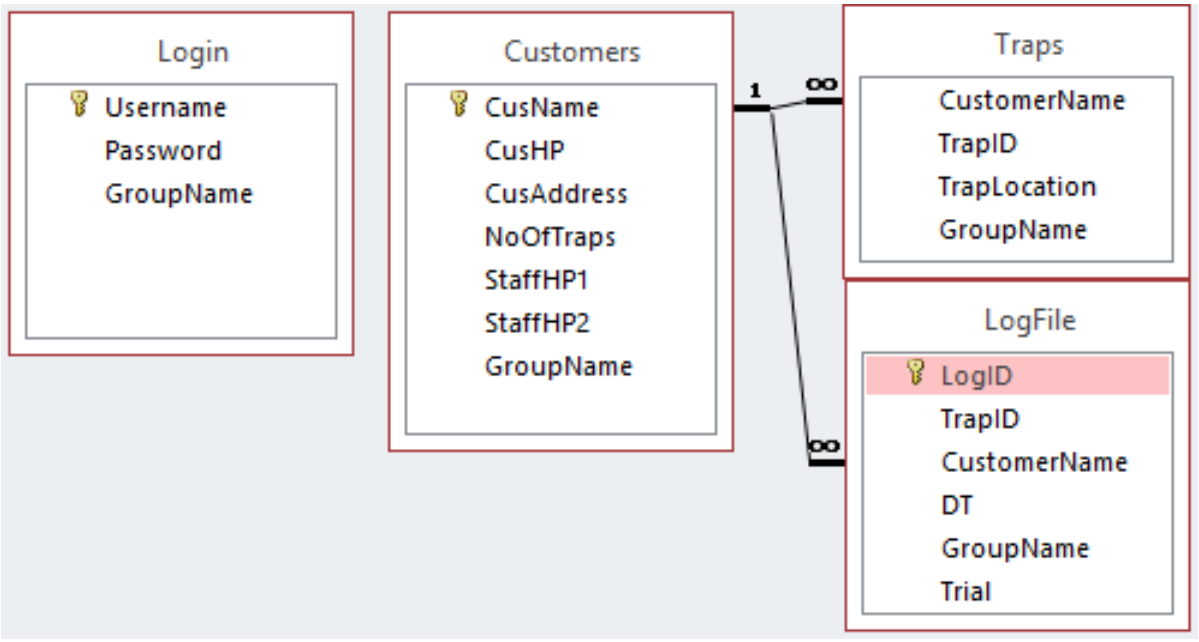


Fig. 27 Relationships between four tables in Access

After the access SQL is exported, import it in MySQL via phpMyAdmin. This would allow MySQL to have the same database as the one in Access. If you are using someone's external database but do not have access to cpanel, with his consent you can remotely access MySQL via a remote client (e.g. HeldiSQL). HeldiSQL is an alternative when you are not able to get into phpMyAdmin.

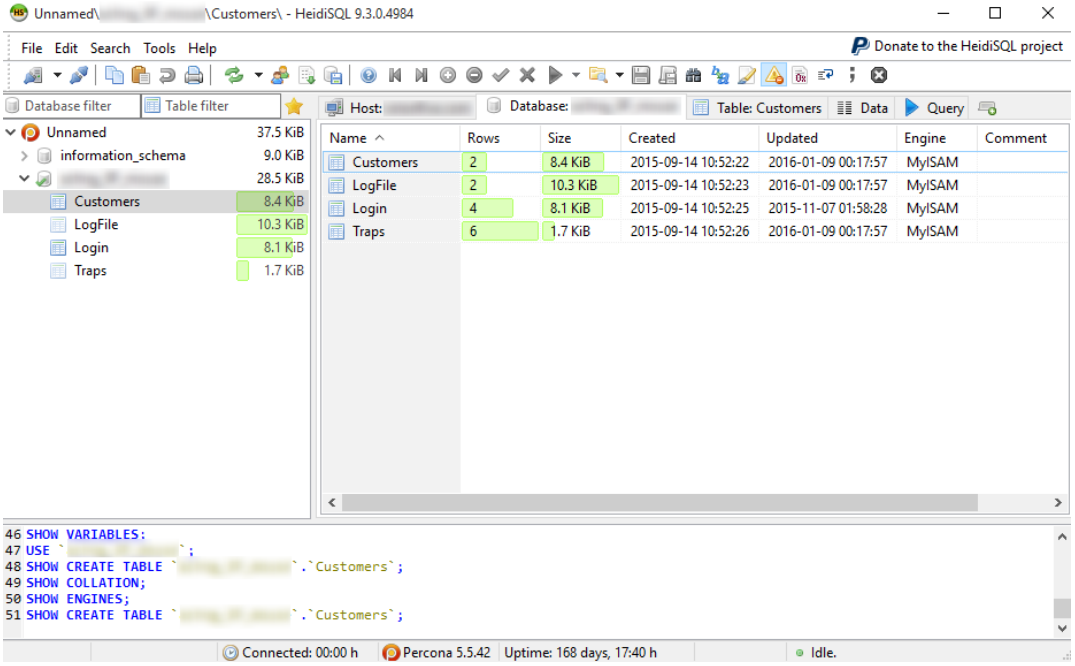


Fig. 28 External database tables in HeldiSQL

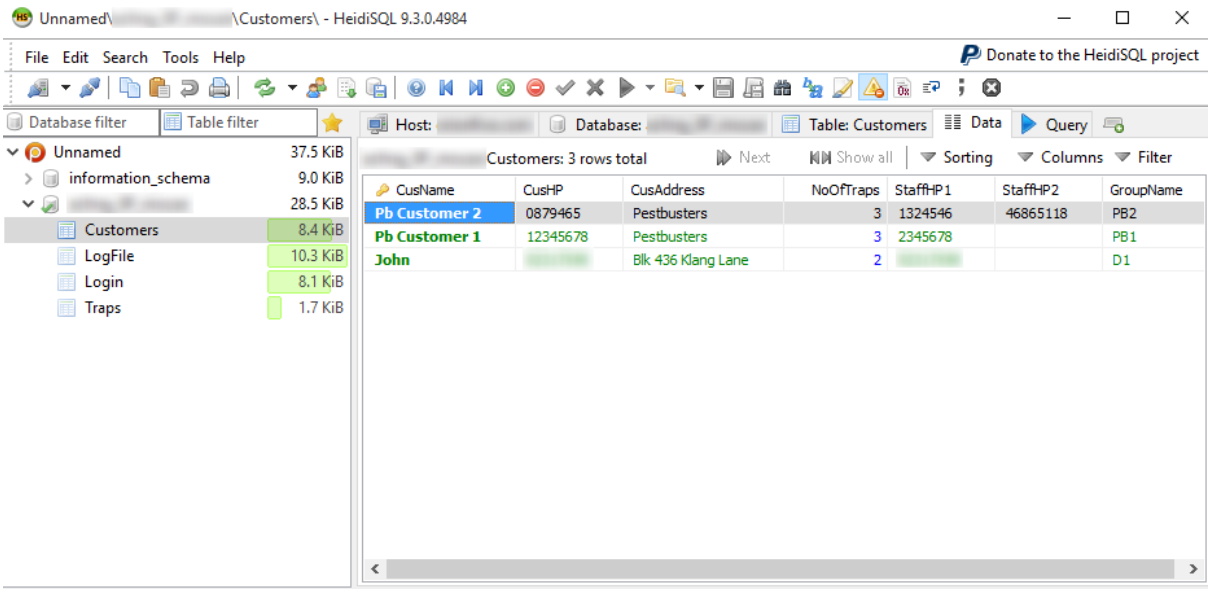


Fig. 29 Customer details in HediSQL

Before we received the external database from the company, we tried creating one from the free hosting servers. They provide one free MySQL database and phpMyAdmin access. Before we start updating the database with information, we need to create tables. Since we had exported a file from Access, we imported it into the database in phpMyAdmin. The tables were created automatically after a successful import. We tried updating the tables first

by inserting SQL queries before we start updating them via the application. This is how it looks like in phpMyAdmin in a free hosting server.

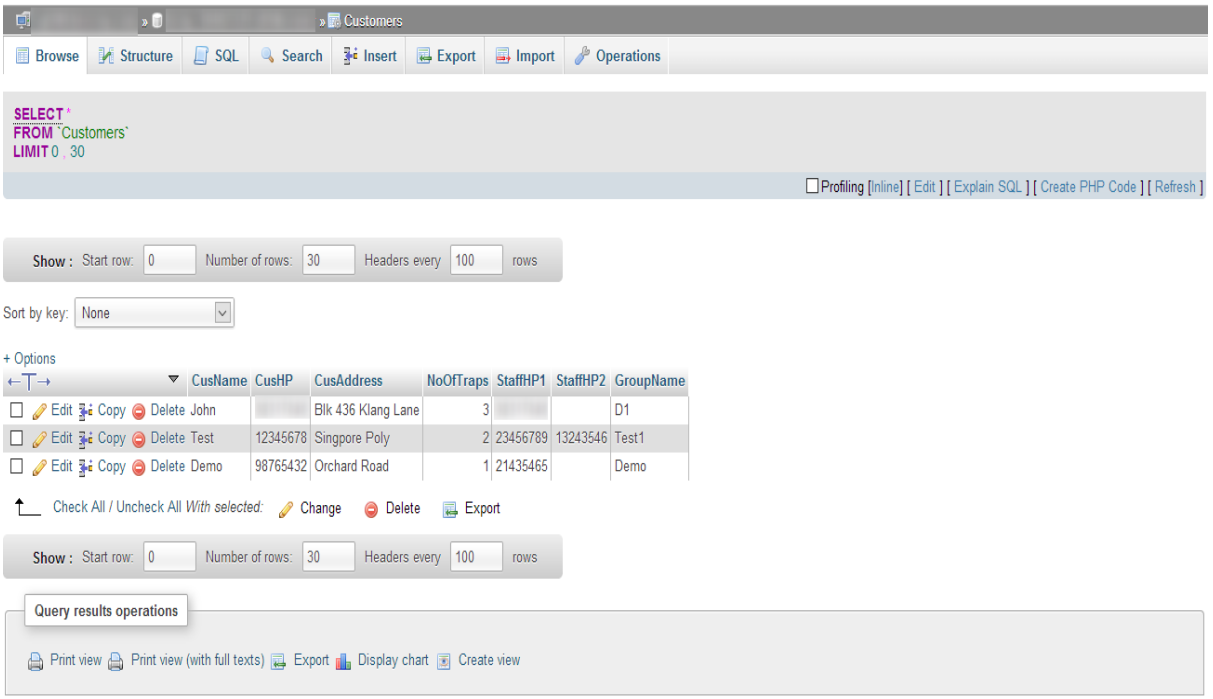


Fig. 30 Customer details in phpMyAdmin from free hosting server

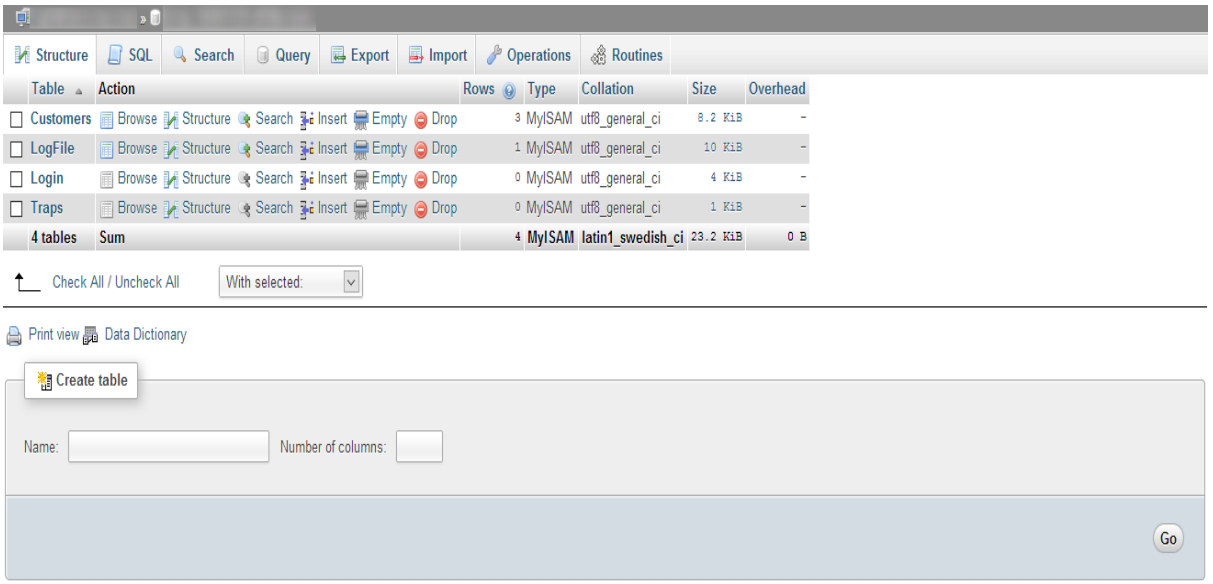


Fig. 31 Database tables in phpMyAdmin from free hosting server

2.4 Features of Mouse Trap Alert System

There are some features of the system that makes it different from the rest in the market. Here are some of the following:

- Instead of designing a new mousetrap, we are focusing on the management of

monitoring the traps. Thus the slaves we had built act as an attachment to the existing spring mousetraps. It is easy to mount, no other special tools needed.

- It is easy to setup the mousetraps. This means the staff do not need to study on how to use the slave properly to make it work. It is just a push of a button as an additional step for setup.
- Instead of using computers/servers/modems to monitor the traps, two smartphones are used to replace all these hardwares to lower the cost and to enable portability. (Without external database server, only one smartphone is required)
- User-friendly interfaces of the applications allows the user for example, to test the connectivity between phone and slave before live monitoring, and to monitor the status of the mousetraps remotely when the user is away.
- Using SMS messaging to alert the staff (Up to two currently) when any of the mousetraps is triggered
- Traps do not need to change for up to 2 months on 3 AAA battery as the attachment device will be remained off all the time until it is triggered. (Rechargeable-batteries are highly recommended)
- Allows the user to see the progress of the trap status, analyze mouse hotspots and plan for redeployment more efficiently
- It is suitable for outdoor monitoring, as the exterior is water-resistant to a certain extent.

IMPLEMENTATION

Week 1 (2.3.15 ~ 6.3.15) : Start of March holiday ITP/FYP - Identify problem faced by PestBusters. Mr Wong (Supervisor) and FYP group discussion on the concept of the solution. Research on learning how to use Arduino and Android Program.

Week 2 (9.3.15 ~ 13.3.15): Mr Wong borrowed Arduino Development Kit to us and start testing of component. Write simple on how the code flows. Drawn GUI and ideas on papers for the interaction between the user and the applications.

Week 3 ~ Week 5 (16.3.15 ~ 3.4.15): Started working on the code for Arduino and Android for showcasing to the industry on week 6. Mr Wong helped us a lot through this 2 weeks.

Week 6 (6.4.15 ~ 10.4.15): Completed first prototype, met our first Pestbusters employee - Mr Afendi, scheduled for showcasing on wednesday for us. Mr Wong and FYP group went together to PestBusters office for showcasing to Mr Thomas Fernandez, CEO of PestBusters. One week later, he decided to sponsor us the hardware components and delivery it to school.

** 8.4.15: Officially, FYP EEE-Group 15A155: Broke up from 3 members to 2 members. One was pulled out from group by Singtel for their FYP project.

Week 7 ~ Week 13: ITP over, school starts. Discussion over the lowering of the cost on slave. We found out about about the new Arduino Board - Arduino Fio v3 and took a huge risk to change the necessary hardware components. Mr Terence Kok, director of Orion Five, branch office of PestBusters which handles smart system came to return the first prototype.

Week 14 (1.6.15 ~ 5.6.15): Start of June Holiday, Arduino Fio v3 delivered to school and ready to review its spec. changing of arduino code and external components. Designing of PCB board. Doing slides for Assessment 1 next week. Range test on the Xbee outfield.

Week 15 (8.6.15 ~ 12.6.15): Start of Assessment 1 on 9.6.15, mention for using Arduino Uno as the microcontroller to our system even though we are changing it to Arduino Fio v3. Soldering of components to the copper board instead of PCB board. Gotten some pair of rainbow wire in school to do 1 sample trap for PestBusters to follow the schematic diagram and testing.

Week 16 (15.6.15 ~ 19.6.15): Searching for Y-cable from Sim-Lim square, Final touch up, Terence came back to collect all the Components to do testing on their office together with the APK file for the phone. Ting Wei did repeater code for arduino and application but were unused and thus scrapped.

Week 17 ~ Week 27 (22.6.15 ~ 4.9.15): Holiday ended, School starts. First week gotten a call from Terrence that the collected prototype not working and wanted us to head down to their office personally to troubleshoot and finish on that day and also found out that Y-cable not working and have to specially buy it online. Gotten a call from Mr Afendi about the feedback on fio prototype, which he suggested us to include a logging server. 1 week before the Sept. holidays, I (Jia Chen) went ahead to try do development of the 2nd App for logging started using 3G/wifi.

Week 28 (7.9.15 ~ 11.9.15): September Holiday starts, we did the interface and the communication using free web server hosting site to do the communication, with account log-in and data display on it. Finished within 2 weeks. Also found out that Mr Afendi was no longer in PestBusters.

Week 29 (14.9.15 ~ 18.9.15): Went alone to School to meet up with Mr Wong and got reprimanded for not informing the updated of the system. Now have to redo the communication and have to burden Ting Wei to change the whole 2nd app.

Week 30 ~ Week 32 (21.9.15 ~ 9.10.15): Ting Wei further updated the second application to receive SMS and then updates the database, while the main application SMS the information to the second application instead of just main application updates the server via WiFi/3G.

Week 33 (12.10.15 ~ 14.10.15): Preparing slides and showcase the Final Prototype for demo

of Assessment 2

Week 34 (19.10.15 ~ 23.10.15): Gotten the online bought Y-cable from Terence and Test for one last time before sending the Final Prototype design to PestBusters.

24.10.15: Officially ended the development of the Prototype.

PROBLEMS ENCOUNTERED (SOFTWARE)

During the creation of the applications, I (Ting Wei) have encountered quite a number of problems. The first major problem is that my first idea is to add mouse trap locations by using a textbox and a '+' button. It is done by pressing '+' button to create another textbox and then fill up that textbox with the locations. After doing it for a while, I felt that it is very complex to create and I have a hard time figuring out how to extract the locations from the newly created text boxes since they are created in the application. Furthermore, the traps have their ID, so I do not expect the user to type e.g "M001-" before writing down his/her location. I researched for other alternatives and found out that listview is suitable for storing locations and spinner is suitable for the user to select the trap ID. These two are excellent in cooperating just one textbox for adding trap locations.

Next, the second major problem that I have also encountered is after the change whereby the slaves will only be powered on when the mouse trap is triggered. Before this change, the system uses polling method. It means that the phone requests for the slaves to reply their statuses. This works well when the slaves are powered on at all times. However, after the change has been made to lower down power consumption, the polling method is not working properly.

The first reason is the slaves are reprogrammed to sleep for an interval of an hour after waking up for about 10 seconds the moment the trap is triggered. This gives the polling method very little time to get all the trap statuses. The second reason is the dongle cannot take in multiple slave replies at the same time, thus the application is programmed to poll one slave at a time. The third reason is that the user has to match with the polling sequence e.g. trap 001 has to be turned on first before the polling of 001 starts, then trap 003 will be turned on next before the polling of 003 starts. The user also have to take note that every trap only wakes up for 10 seconds. Thus the more traps deployed, the harder and longer polling will be. The solution to this problem is that instead of using polling, it is using event triggered. This means that the phone is just waiting for the slaves to send their statuses when triggered. Even though it is not as reliable as the polling method, it is simple and easy for the user to test the slaves.

The third major problem occurs when adding traps ID and their locations to the external database. In the early development of creating the second application, its purpose was just to

display the status of the traps from the external database. The main application was the one that updates the external database via WiFi/3G. Adding customer details to the external database is easy, however for adding traps ID and their locations, it requires many SQL queries as one query is used for one trap ID and its location. To have that many SQL queries, the application has to send one by one, which I do not wish to resort to that. So I researched on adding all the traps ID and their locations by just one SQL query. I have also created an algorithm to combine all traps ID into one string and their locations into another string in the application before separating them in PHP for SQL query.

Next, the feedback from Terence is that the power consumption will be high if the phone turns on WiFi/3G for the main application to update the external server. This is another major problem as the phone with the dongle is powered by the power bank (no power source nearby), thus it may not last long to monitor the traps. SMS is a viable option as it is very reliable although it has a flaw, which is the character limit. This flaw results in many problems.

On top of it, the second application that receives the SMS messages will not know which SMS is for which table to update in the external database. I had to think of how to differentiate the messages so that the second application can recognise them and updates the table correctly. For example, to log the traps being triggered, the main application sends with “LM” at the front to indicate that this is a log information. And to remove customer it uses “RC” at the front of the message.

In addition, I had to think on how to add multiple traps with either single or multiple SMS messages. If there are multiple SMS messages since one is not enough, an algorithm is created to add another “AT” (indication of adding traps) at the front of every SMS message.

Another problem is that the second application will not know whether there is only one or multiple SMS messages to update traps table. For multiple SMS messages, this results in the second SMS traps information overriding the ones in the first SMS and the third overrides the second since the PHP is coded to remove existing traps information before adding the new ones from the SMS. Therefore, I have decided to include either ‘T’ or ‘F’ (true or false) at the end of the SMS to indicate whether this SMS is the first. If it is the first SMS, the existing traps information will be removed and the new ones from the first SMS are added. If it is the second SMS onwards, the existing information remains and the new ones from the (non-first) SMS are added.

For the SMS configuration, I have used around 300+ SMS messages to make these applications stable and working properly.

CONCLUSION

The Mouse Trap Alert System is a monitoring system that aims to improve the efficiency of monitoring mouse traps and reduce the need of manpower. User-friendly applications such as RatzManifest and RatzManifest Monitor help the staff to set up the traps and monitor them with ease. Using Zigbees that covers a wide area, the staff is able to set up multiple traps within that area.

It is currently completed. The applications are tested thoroughly to be stable. Even though the hardware components especially the slave gave us many problems, they are working properly if there are no faulty components. Till this date, there is still no live-monitoring results by the staff in PestBusters. We will still try to get results from them even after writing this report.

Although we had done many improvements since the start of the project, there are still some improvements that can be made to further improve the system. Here are some of the following:

- Better hardware components (eg. battery holder, wires) to make the slave stable
- Better and higher range of communication using different products instead of Zigbee
- Rechargeability using rechargeable batteries or lithium batteries instead of alkaline or zinc
- Lower consumption for the master if using power bank
- Plotting a graph of a certain period for the mouse traps that triggered most frequently

REFERENCES

1. http://img.directindustry.com/images_di/photo-g/9260-2505015.jpg
2. http://homanndesigns.com/images/RedPushbutton.jpg
3. http://www.mini-circuit-breakers.com/photo/mini-circuit-breakers/editor/20131230145443_49705.jpg
4. https://www.adafruit.com/images/1200x900/754-00.jpg
5. http://www.skycraftsurplus.com/images/view.aspx?productId=4766
6. https://www.futurlec.com/Pictures/DIP5_200.jpg
7. http://shop.valarm.net/collections/cables-connectors/products/micro-usb-host-otg-y-cable-with-micro-usb-power-for-samsung
8. https://www.sparkfun.com/products/11520
9. https://www.sparkfun.com/products/10116
10. http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu
11. http://developer.android.com/tools/studio/index.html
12. http://g02.s.alicdn.com/kf/HTB1nJAmIFXXXXaZXFXXq6xXFXXXY/Single-Core-Copper-Electrical-Wire-2-5mm2.jpg
13. https://s-media-cache-ak0.pinimg.com/736x/3d/94/34/3d9434d1be7c0a6a2941d4b9aa

883c3d.jpg

ADDITIONAL INFORMATION

Hardware:

1. http://jamesreubenknowles.com/level-shifting-stragety-experments-1741
2. http://www.ni.com/tutorial/5631/en/
3. https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu
4. XBee - Broadcast and Unicast addressing modes

Arduino:

1. http://www.toptechboy.com/arduino-lessons/
2. https://www.arduino.cc/en/Main/ArduinoBoardFioTips
3. https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide
4. https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/Arduino-Fio_v31.pdf

Android:

1. http://developer.android.com/training/index.html
2. Login And Register Guide
3. http://developer.android.com/sdk/installing/create-project.html
4. https://www.youtube.com/watch?v=oi9PvZs-dq4

Server Side Scripting:

1. http://www.tutorialspoint.com/mysql/
2. http://www.newthinktank.com/2014/08/mysql-video-tutorial/
3. http://www.tutorialspoint.com/android/android_php_mysql.htm

Documentation:

1. https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Manual.pdf?_ga=1.154433272.1974121935.1425885866
2. http://www.farnell.com/datasheets/27606.pdf
3. https://cdn.sparkfun.com/datasheets/Components/General%20IC/33244_SPCN.pdf
4. http://www.atmel.com/dyn/resources/prod_documents/doc2486.pd