DONGLE DATABASE AND GUI FOR MANAGING UNIQUE BIRD TAD IDS

A Design Project Report
Presented to the School of Electrical and Computer Engineering of Cornell University
in Partial Fulfilment of the Requirements for the Degree of Master of Engineering, Electrical and Computer Engineering

Submitted by
Di Jiang, Jinghan Du, Lei Zhang, Rui Meng
MEng Field Advisor: Joe Skovira
MEng Outside Advisor: David W Winkler
Degree Date: January 2017

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title: Dongle Database and GUI for Managing Unique Bird Tag IDs

Authors: Di Jiang(dj327), Jinghan Du(jd855), Lei Zhang(lz392), Rui Meng(rm879).

Abstract: Our project is an interdisciplinary project including Biology and ECE fields. Our team has 4 members and we are responsible for the ECE part. First of all, we developed a USB dongle containing a receiver to detect tag codes and a GPS receiver to record location. As for the tag part, we designed a new tag with a smaller solar beeper which can utilize the solar energy more efficiently. In addition, we also extended and refined the GUI and its interaction with the cloud-based database.

After the project is finished, field biologists can use this device with a laptop computer to capture tag codes, times and locations as they are deployed, along with species and band number of the animals being tagged. Whenever a tag ID is recovered through RF communication in the field, similar data will be recorded and stored to the cloud-based database. It is a very meaningful project for protecting the endangered birds. What's more, this project can also be applied to other endangered animals.

Personal Contributions：

Jinghan Du:

1. Dongle programming

The dongle need to be programmed to listen to the tag IDs and send those information to the user's computer.

There were several possible solutions for this problem. One of them is to write a completely new function on the board to make the board listen to the signals. Also, the board could transmit all the information it heard in the buffer to the computer or it can send only the tag IDs.

Here I chose to write to code in the default functions given by the board demo. Since the board can listen to the signals at the specified frequency, I just need to let it send what it heard to the computer. This is the simplest way as far as I can see, and it is easy to debug.

As for the information to send. I found there was a buffer to store all the message the board had received. And the information about the tag ID is always stored in a fixed place in the buffer. So I programmed to let it only send those tag IDs to the computer. In this way, the dongle is only transmitting useful information.

2. Dongle layout refinement

The overall size of the dongle must be smaller so that users could take it with them. Thus a new dongle with only the useful parts of the old board (Si1060 and a USB/serial adapter) needs to be designed.

The Si1060 developer board has a lot of elements that we don't need, such as the LED lights. Basically there are two choices about how to simplify this: either we only take out the Si1060 chip or we combine Si1060 and USB/Serial adapter together. The former choice would definitely make the new dongle smaller than the later one. But with a USB interface, the new dongle could be easily tested without connecting to another USB/serial adapter. On the other hand, USB port might not be useful when the dongle is combined with the Raspberry-Pi.

We designed the new dongle with the USB/serial adapter. Because one of the functions of the dongle is to be plugged into the computer to send data to GUI. Thus the USB/serial adapter is indispensable.

3. Tag layout refinement

The current tag is working fine but we can still make changes to its board to make it have a wider range of frequency and a longer signal transmission distance. Achieve this, the board has to utilize solar energy in a more efficient way.

We chose the si1060 RF chip finally. Because in this way, we can use the same chip on RF Dongle. The chip can also support a wide range of operating frequencies so we can use it for the current 434 MHz design and also for tags and base stations that will operate at 166 MHz for the wildlife tracking system. As

for the board layout design, we used double layer on the tag so that those elements can fit into a board as small as a dime.

All of the detailed documentation of design implementation and final results of the above problems are discussed below.

Lei Zhang:

1. Regular expression programming

For the tagID received from the tag and the GPS coordinate information received from the GPS module, we need to check the format of the information before sending the information to GUI. Because we need to make sure the information is in standard format and we need to violate hackers' attack for security reasons. Therefore, we must figure out the way to check the format of the information.

After sincere consideration, we chose Regular Expression to check the format of the information. Because this is an easy and general way to examine the format of data. However, there were several ways to implement regular expression, for example, we can program on the Dongle part and check the information after receiving the information to Dongle, and then send the information of good format to GUI. We can also program on the GUI part to examine the format.

In order to improve the efficiency of our system and make our system more robust, I chose to program on the GUI part, because programming on software part is more stable and easy to debug. In this way, our system is much more robust.

And I parsed the tagID and GPS coordinate into several parts according to the standard formats, separating them according to the position of numbers or letters. After the Dongle received the information, we will use the regular expression to examine the format and only send the good-formatted information to GUI.

2. Dongle layout refinement

We combined the si1060 and USB/Serial adapter together to make the Dongle, however, the Dongle is too big in this way, and it is not easy for ecologists to take it to the fields. So we extracted the most useful parts from the old Dongle and then designed the new Dongle with smaller size.

There are two ways about how to simplify the Dongle, one solution is that we only take the RF chip on si1060 and abandoned the USB part, in this way the Dongle will be much smaller. The other way is to keep the USB part and the Dongle will be a little bigger. After cautious consideration and heated discussion, we chose to keep the USB part. Because it is easy to test the Dongle with a USB interface, we do not need to use other USB adapters. Also, the users of the Dongle can simply plug the Dongle in their laptops easily. What's more, when we combine the Dongle with the Raspberry-Pi, the USB port is convenient as well.

3. New tag layout design

The original tag works pretty well at present, but we want the tag to listen to a wider frequency and transmit RF signal in a longer distance as well. So we decided to design a new tag.

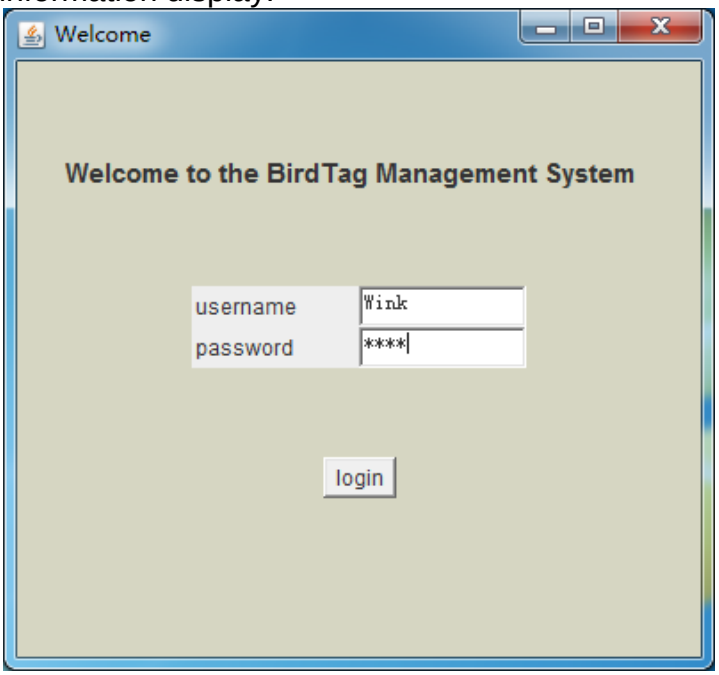
Di Jiang:

1. Design and program from the prototype of GUI

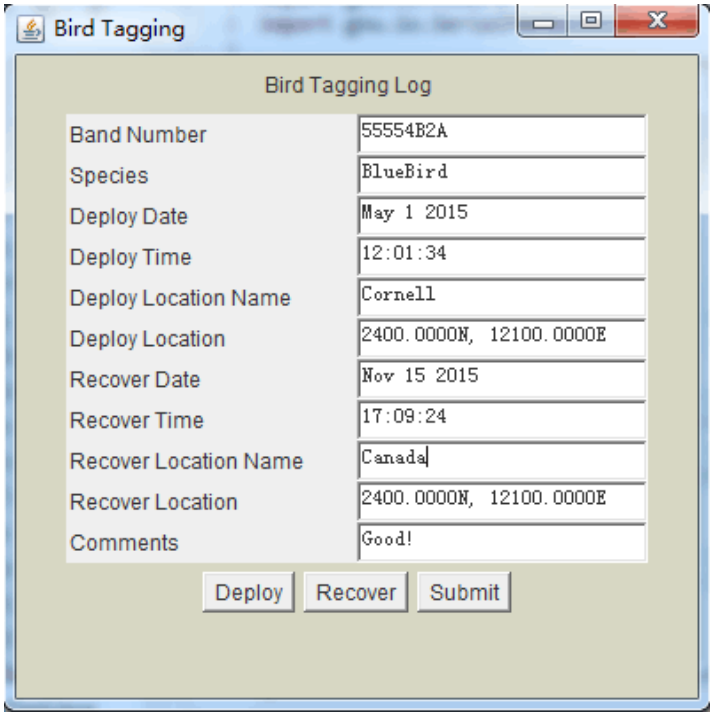
The first task I received for this project is creating a GUI for bird tag information management. I construct two basic interface to implement login and information display function.

I used simple label, textField and button to construct the interface, FlowLayOut to display these components.

The login interface could parse the username and password and switch to information display.



The information display interface has three buttons, both display and recover buttons could receive the GPS information and tagID from GPS module and Si1060, check the format of information and show them in GUI. Submit button could save the information in GUI to a txt file since our team don't have a server to save information in cloud database.



2. Receive and display GPS information and tagID on GUI
When our team could receive tagID from Si1060 and display the tagID in the board, I start to display this information and GPS information from GPS module.

Firstly, I use realTerm to test whether I could receive information from dongle and GPS module. Secondly, I write methods using JAVA to open the two ports which connect GPS module and dongle respectively, read the information from two port alternatively.

Thirdly, I started to process the data I got. GPS information comes in a format of string starts with GPGGA, I parse the GPS information from the long string and using regular expression to check whether it's a valid information and transmit it to GUI textfield.

Last, I get local time and date from the computer and display them in GUI too when Deploy and Recover button are clicked.

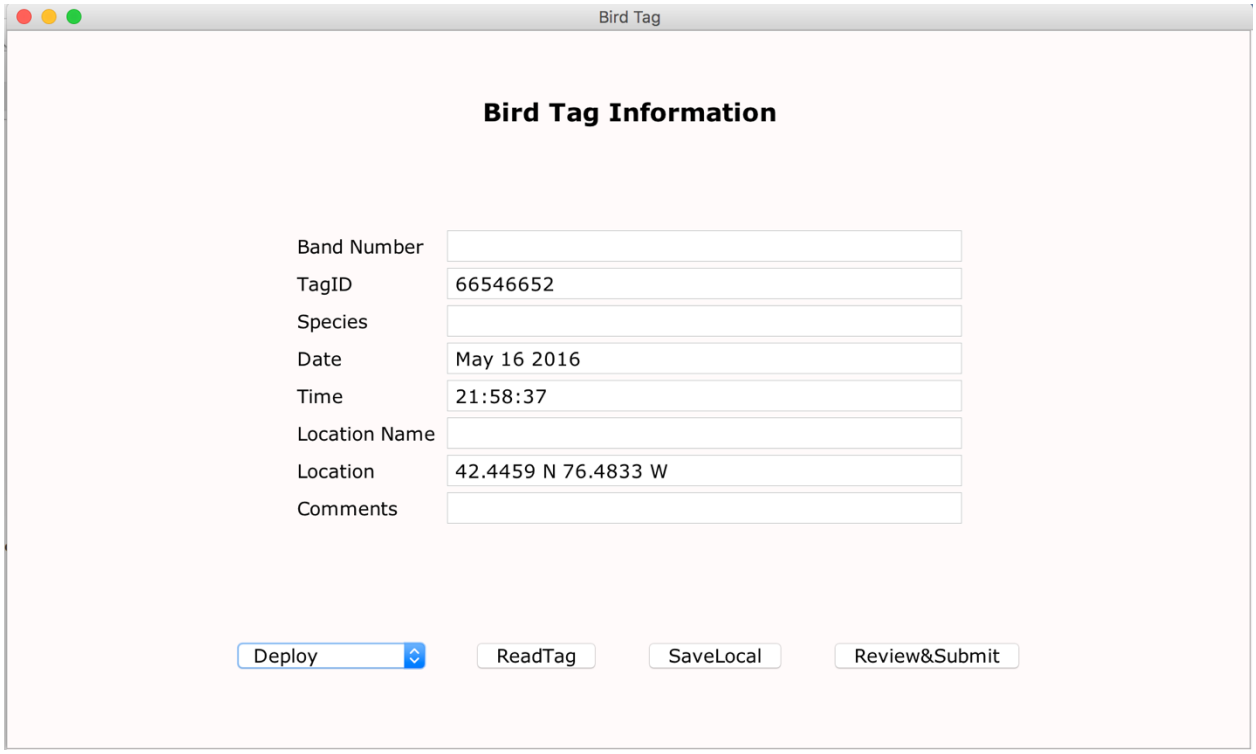
When submit button is clicked, I get the string from all text field and combine them to a long string and save them in a local txt file.

3. Test latest version Dongle
In the spring semester, our team receive the latest Dongle and I test it with Rui. We programmed and downloaded the code to Dongle and connected Dongle with computer. We used realTerm to test whether Dongle could transmit data and redesigned the connection part with GUI, after fixing bugs with latest Dongle, I started to work on refining GUI.

4. Design and refine GUI according to user's needs

I refined GUI from four aspects, styling, information checking, data management and database connection which I have described in Part 3 GUI and Database Connection.

In styling, I pick new font, background color, different layout to reconstruct the interface. In information checking, I step to expand regular expression, parse each information a particular checking format and pop up an alert window if the information is invalid. In data management and database connection, I construct temp txt file and permanent txt file to achieve these functions.



Rui Meng:

1. Transmit GPS and tagID information

For Jinghan Du has already successfully transmit GPS and tagID information to laptop, the problem needs to be solved in this part is to transmit these information to the GUI.

Because the stream transmitted to the laptop includes information we need and other things we don't need, I need to find a solution to extract the useful information. The possible solutions are to extract the substring or to combine this with regular expression in JAVA.

After testing the two methods, I choose the latter solution for using regular expression can make sure the information are in right format we want so it's more secure. The specific solution in JAVA is to use substring method and

Pattern and Matcher class in JAVA. The detailed documentation of design implementation and the test result is discussed in 3.3.2.

2. Dongle layout refinement
The problem needs to solved in this part is to make a smaller dongle so that it will be easier for the biologists to take it with them.

The solution to the problem is to discard the elements we don't need in Si1060 developer platform and just keep the Si1060 microcontroller. Then we combine it with a USB adapter. The detailed documentation of design implementation is discussed in 2.2.2.

3. Sample dongle test
After the schematic and PCB design, we need to test the sample USB dongle to make sure it can detect the tagID information as well as connecting to the laptop. The test method and result are discussed in 2.2.3 and 2.3. Now the USB dongle has been put into manufacture.

4. GUI refinement
The problem in this part is to let the biologists to get tagID and GPS information in the GUI and also enter other fields. After checking, these information should be sent to the cloud database.

For complete information, we add different text fields; to let the user read tagID, GPS and time information, we add "readtag" button; to make sure the information entered into database are secure, we add the function to check the illegal input; to provide users the chance to correct the typos, we add the review and edit interface; for the consideration of that users may not get access to the Internet all the time, we add the function to save entries to local and check the Internet connection...The detailed documentation of design implementation and test result are discussed in Section 3.

Executive Summary

GUI:

1. Accomplishments of GUI:

1. Starting from a simple version of GUI, we programmed on it to receive tagID and GPS information from the Dongle and GPS receiver and then show them on the textfield.
2. We refined the GUI by adding the areas in the login interface, modifying the fields according to user's need in the bird tag information interface.
3. We validate user inputs in the bird tag information interface to avoid illegal inputs.
4. We built the third interface of GUI for users to review and submit. The user can edit any field of the entries entered by himself/herself.

2. Challenges of GUI:

1. The user needs to get sufficient information from the GUI including the tagID, GPS and time.
2. The user should enter information easily and be able to review and edit the entries he/she entered.
3. Before submitting to the database, all the fields should be checked to avoid illegal input.
4. The GUI should check the Internet connection and if no connection save entries to local.

Dongle:

1. Accomplishments of Dongle:

1. We programmed on Si1060, a RF microcontroller manufactured by Silicon Labs on which we programmed on to let it listen to certain radio frequency, get the tagID information and then transmit to the laptop constantly.
2. We drew the schematic to connect the crucial part of Si1060 with USB port adapter and arranged their location in the Eagle file, which includes the schematic and the PCB layout of the board, to package them into a USB Dongle and sent out for prototype.
3. We tested the sample and found the problems. After testing, the USB Dongle is put into manufacture. The final version of the Dongle is small, convenient to use and works correctly!

2. Challenges of Dongle:

1. The Dongle should listen to the RF signal and receive the tagID constantly. When there are multiple tags, it should detect all of them among the range.
2. The Dongle should be able to transmit the tagID information into laptop.
3. The Dongle should be as small as possible. It should be a plug-in device for biologist's convenience.

Tag:

1. Accomplishments of tag:

1. We changed the RF chip. The new chip can receive, not just transmit, so we can use the same chip on the RF dongle. The new chip also supports a wider range of operating frequencies so we can use it for the current 434 MHz design and also for tags and base stations that will operate at 166 MHz for the wildlife tracking system.
2. We designed the tag using Eagle file, including the schematic and PCB layout.
3. We sent the design file to factory to manufacture the tag and the first version is under testing.
4. The tag's size is really small, it is just as big as a dime, and the tag can be attached to a bird like backpack easily. It is really flexible and will not hurt the birds. Besides, the tag can also utilize the solar energy efficiently.

2. Challenges of tag:

1. Since the tag is attached to birds, it cannot be too big or heavy in case that the birds will be affected.
2. The solar chip should utilize solar energy efficiently.
3. The RF chip should listen to a wider range of frequencies.
4. If the distance from Dongle or base station is too far, the RF chip could not send signal.

1. Total View of the project

Our system is composed of Dongle, GUI and Tag. And the specific implementations of each part are explained as follows.

2. Dongle

2.1 Objective

The main object of Dongle is to receive tagID information. When a bird with a tag fly by, we can get the tagID of it. Thus we can track the birds.

Here we used Si1060 as the dongle board. Silicon Laboratories' Si1060 Wireless MCUs combine high-performance wireless connectivity and ultra-low power microcontroller processing into a small 5x6 mm form factor. Support for major frequency bands in the 142 to 1050 MHz range is provided including an integrated advanced packet handling engine and the ability to realize a link budget of up to 146 dB. The devices have been optimized to minimize energy consumption for battery-backed applications by minimizing TX, RX, active, and sleep mode current as well as supporting fast wake-up times.

Table 2.1 Si1060 Product Features

Orderable Part Number	Radio	Flash	RAM	DC-DC Boost	Frequency					Max Output Power	Max Data Rate	Sensitivity		Advanced Features*
					142-175 MHz	283-350 MHz	425-525 MHz	850-960 MHz	960-1050 MHz			Max	40Kbps, GFSK	
Si1060-A-GM	EZRadioPro	64 KB	4 KB	No	✓		✓	✓	✓	+20 dBm	1 Mbps	-126 dBm	-110 dBm	Yes

First we need to program on the Si1060 board and let it to listen to the RF signals and transmit the tagID information constantly. Then we will draw the PCB and design the layout. For the final goal, we need a USB Dongle. It can be used both on base station and on biologist's laptop. So when the biologists go to the field, they can bring the USB Dongle with them and use it to track the birds conveniently.

2.2 Detailed design

There are mainly three parts in Dongle part: programming on Si1060, designing USB Dongle layout and sample testing.

2.2.1 Programming on Si1060

To receive the tagID information, first we implement the program of the Si1060 part in the IDE platform. The Si1060 board can keep listening to the RF signals and send the tagID to laptop once it detects the signal with entire ID information.

To achieve this, we did the work in several procedures. First we need to make sure the Si1060 evaluation kit can detect GPT transmissions. So followed by the instructor, we did several configurations and experiments, including setting up the right format of the receiving packet and baud rate based on the developer kit of the Si1060 evaluation kit. At first we only tested one tag with a fixed tagID and after the Si1060 can detect it, we made other configurations to make sure it can hear multiple tags.

When a tag is detected, the board should be modified to convert the binary tag ID code to a character string and send it out on the USB serial port. This could be divided into several experimental steps. First we learnt about the mechanism of how the tag ID is detected and stored in the board. After finding the buffer, we use “Add to Watch” to see the information in the buffer while the board is receiving tag ID. To make sure we are sending out correct information from the buffer, first we let the screen display the information that we are sending out. Figure 2.1 is an example of the board displaying the tag ID.

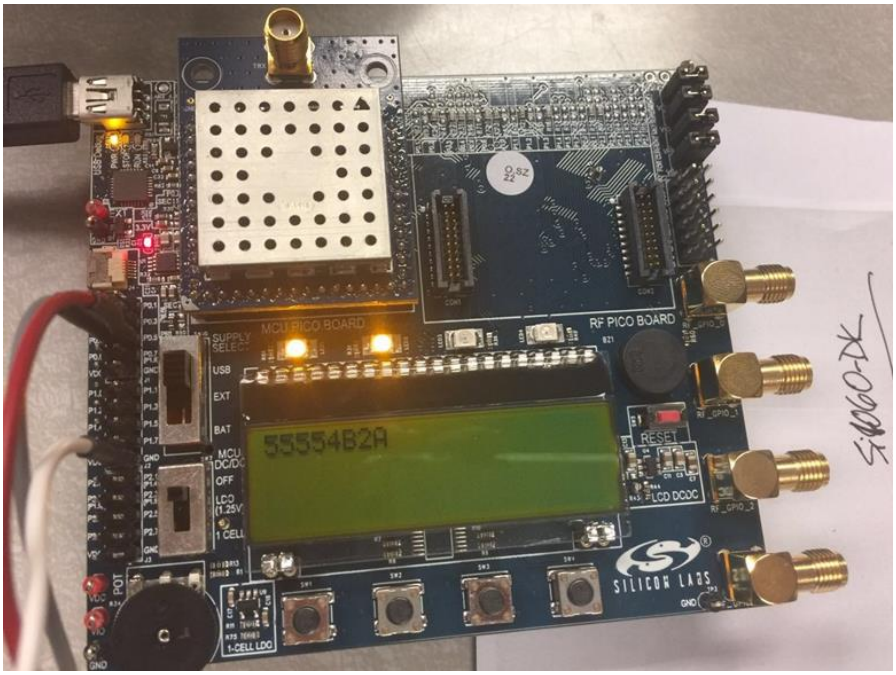


Figure 2.1 Si1060 is displaying the tag ID it received.

Now that we can send the tag ID to computer in a similar way. But before that, the Si1060 should be connected to a serial I/O. Through that it could receive RF tag ID and send it to computer.

2.2.2 USB Dongle Layout design

Now the Dongle can receive the tagID information. But the Si1060 board is too big for base station or laptop. Instead, we need a smaller version of Dongle for convenience and economic consideration. The Si1060 developer platform has some parts like LED screen we do not need. The function of detecting the tagID information is implemented by the Si1060 microcontroller. So we can just keep this part.

We decide to design a USB Dongle. It will consist of USB serial port and Si1060 microcontroller. In order to manufacture the Dongle, we need to combine the 2 parts together and then manufacture the Dongle. The Si1060 microcontroller is extracted from the Si1060 developer platform. Then we work on the USB connector on Eagle. For the USB serial port part, we used USB/serial chip (FT232RQ) and use the USB-A-SMT-MALE variation. We studied the data sheet of the USB serial connection to the MCU and in order to make the connector more suitable to our board design, some modifications were made according to the data sheet, like removing the 3-pin

jumper and connecting VCCIO directly to 3V3OUT. Then we draw the schematic to connect the 2 parts together like below:

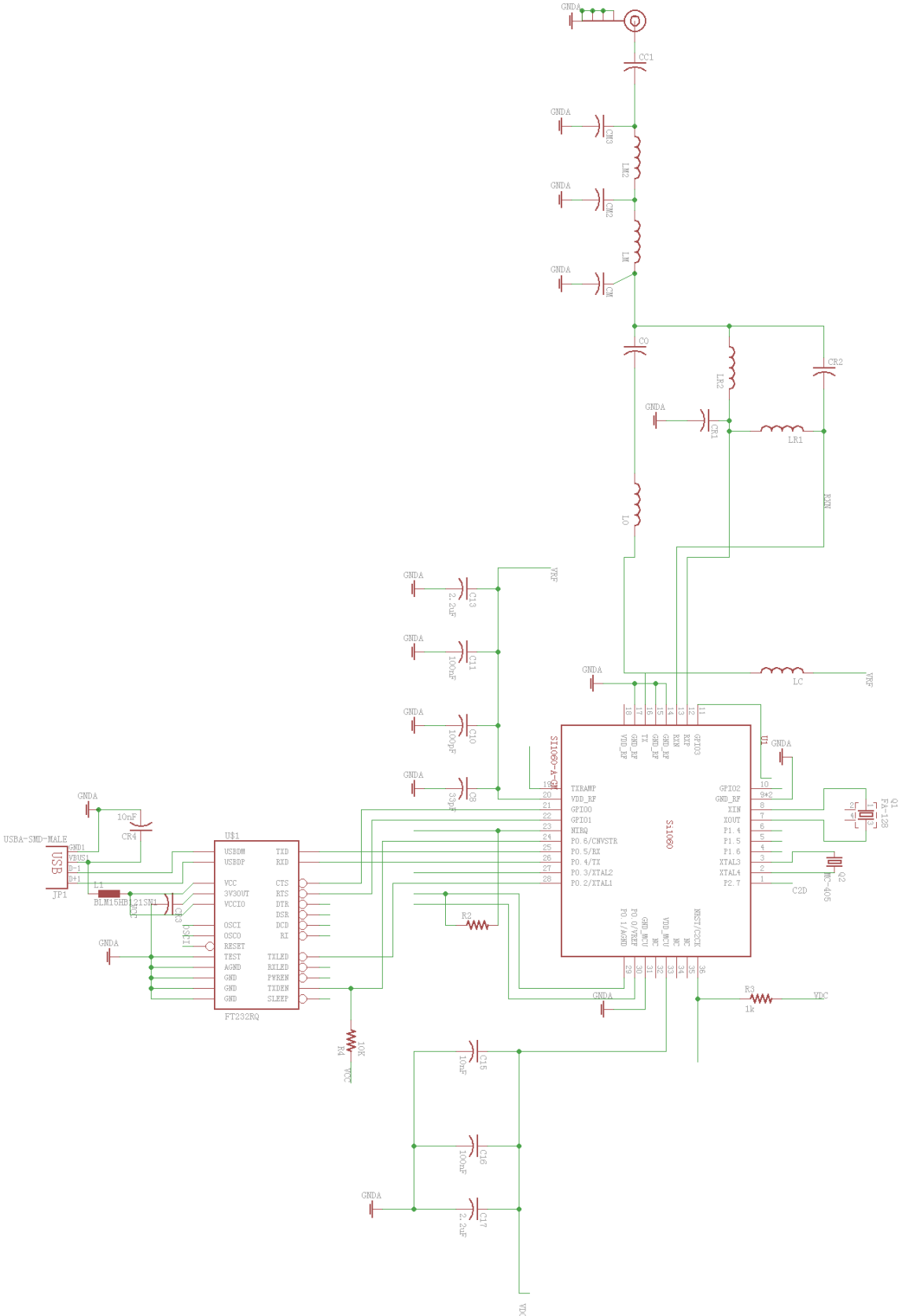


Figure 2.2 dongle schematic

Then we arrange the components and connect them with wires according to the schematic on PCB. We use double layer and try many methods to arrange them more reasonable in order to make the whole package as small as possible. For the contour of the PCB, we use the MINI-REMOTE-OUTLINE (package). We place the USB connector at one end of the PCB outline so it hangs over the edge. The PCB layout looks like below:

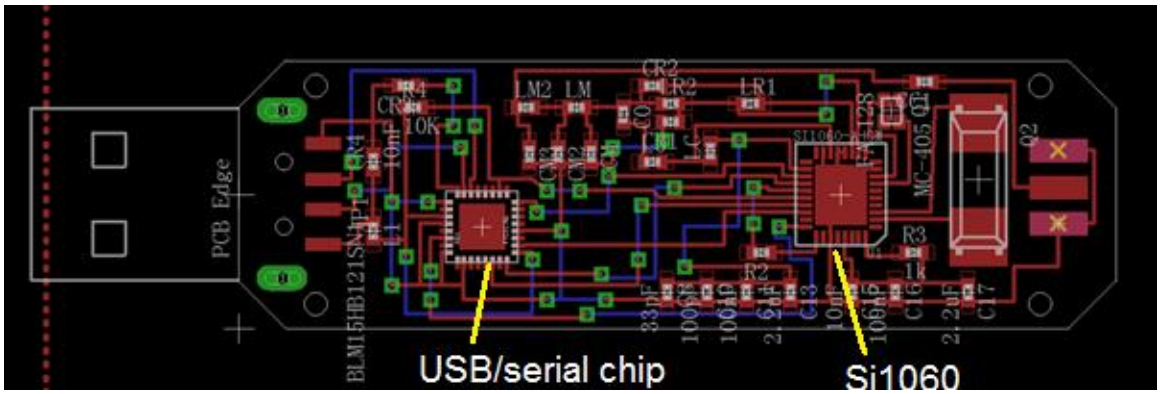


Figure 2.3 Dongle PCB layout

2.2.3 Sample test

After PCB design, first we need to get one sample and test on it. Rich Gabrielson, a visiting scientist and also one of our instructors, and other members of TABER team sent the design file to factory to manufacture and we are responsible for testing the sample and trying to find the problems. We download the Si1060 program mentioned above into the sample, connect it to our laptop and use the Realterm (a terminal software to capture data) to read the information. At first, we could not read the information. We tried to use the Si1060 evaluation kit at the same time and found that it could still read the information. By that we ruled out the possibility of tag problem. So the problem is most likely to be the connection problem between Dongle and the laptop. Then Rich and we used the multimeter to test the pins and wires and found the problem. Rich and other members in TABER team fixed those problems and we tested it again with Realterm and then with our GUI to make sure the sample could receive one or more tagID constantly and transmit those to our laptop. After that we order more USB Dongles.

2.3 Results

Now we get the final version of the USB Dongle. The picture below shows it:

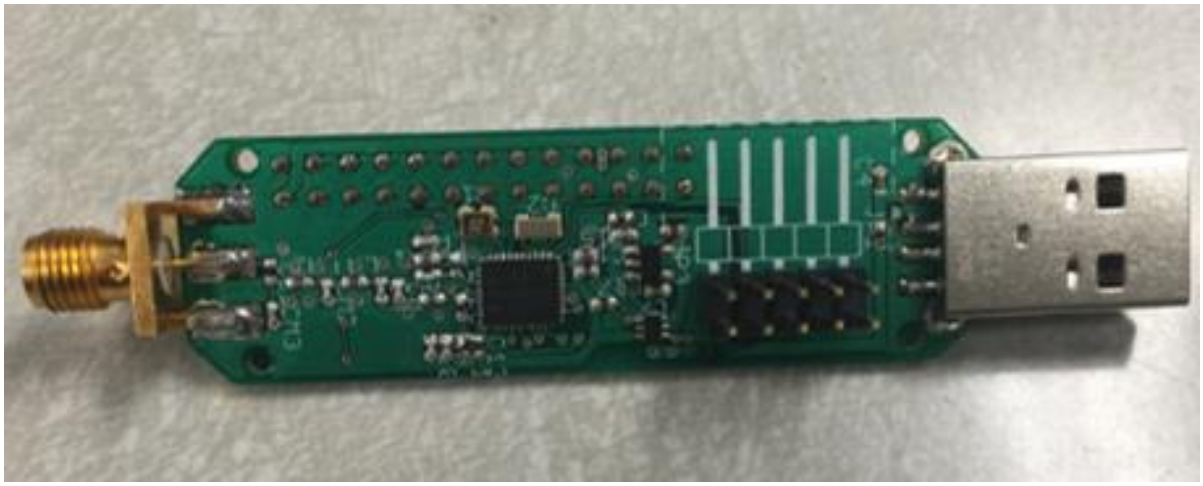


Figure 2.4 final version of Dongle

As we can see from the picture below, the final version of the dongle is much smaller than the previous Si1060 board version. Biologists can take it with them when going to the field and track birds easily by plugging it into the laptop.

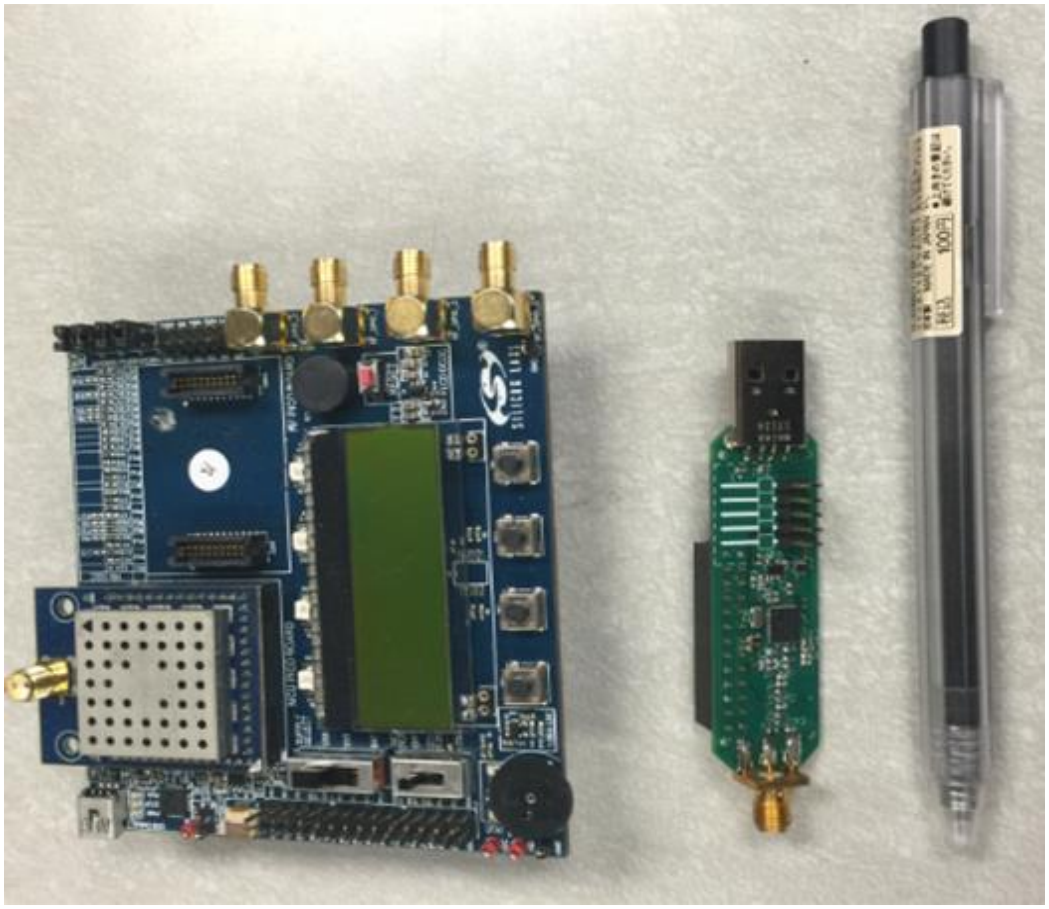


Figure 2.5 first and final version of Dongle

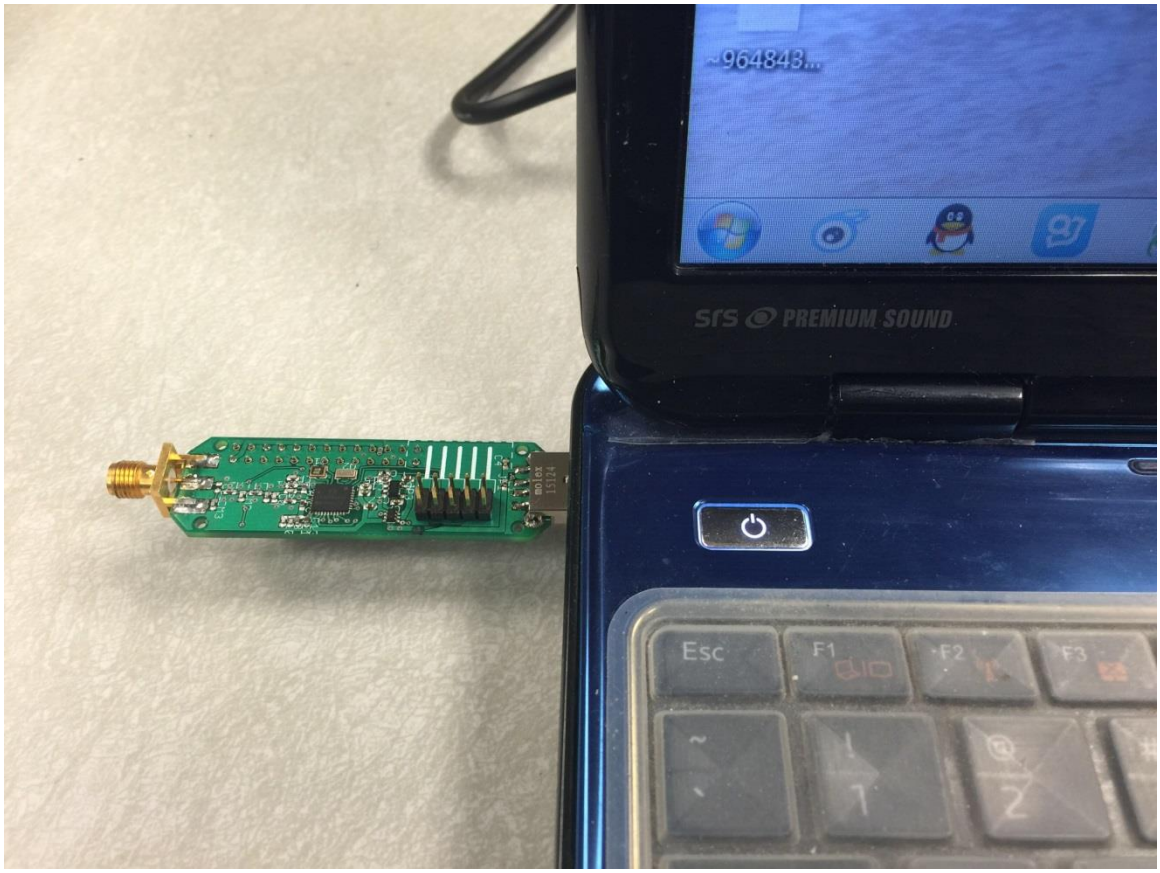


Figure 2.6 The new dongle is much easier to use

It can also be used with the Raspberry-Pi in the base station. Another member of TABER team is updating the Raspberry-Pi so that there be a base station with the dongle combined with the Raspberry-Pi.

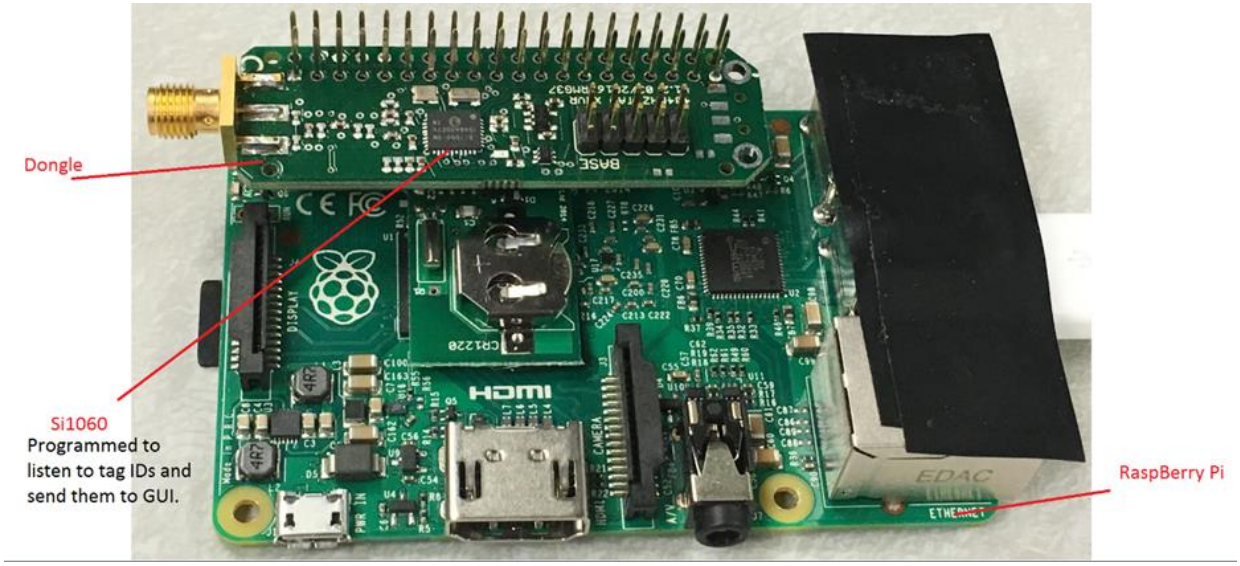


Figure 2.7 Dongle with RaspBerry Pi

3. GUI and Database Connection

3.1 Objective

GUI and Database Connection consist of four sub-tasks, GUI implementation, Information Receiving, Local Information Storing and Cloud Database Connection. Our goals are:

1. In GUI implementation part, we need to construct a user-friendly GUI interface according to the biologists' requirement.
2. In Information Receiving part, user should receive tag ID from Dongle, GPS information from GPS module and current Date and Time for convenience.
3. In Local Information Storing part, the program should save all the information from the GUI information, check the format for each information for database security and save the information in local computer permanently.
4. In the Cloud Database Connection part that needs program to check internet, provide an interface for user to edit information and send data to cloud database.

3.2 Detailed Design

3.2.1 GUI Implementation

We use java to implement GUI and Database Connection and construct three interfaces to achieve Biologists' requirements:

1. Login page with specific username and password.
2. A main page to choose different location and bird status, receive information, add comments, save information in local computer.
3. A review page to double check the input data and edit information. And submit the information to cloud database.
4. The GUI program can protect the server security from hackers by checking the user inputs. Only characters, numbers and several punctuations are acceptable.

The first interface provides text fields to enter username and password, a dropdown menu to choose location, and login button. The different areas will affect the format of the Band Number.

Figure 3.1 Login Interface

Figure 3.2 Choose Region

The second interface uses eight text field to show Band Number, Tag ID, Species, Data, Time, Location Name, Location and Comments; a dropdown menu to choose bird status; a ReadTag button to receive Tag ID, Date, Time, Location information ; a SaveLocal button to check whether the information in GUI in the correct format, it will pop up an alert window to indicate which information is not correct, and save all the information to a txt file in local computer with previous information which are not send to cloud database; a Review&Submit button to pop up the third interface to check information.

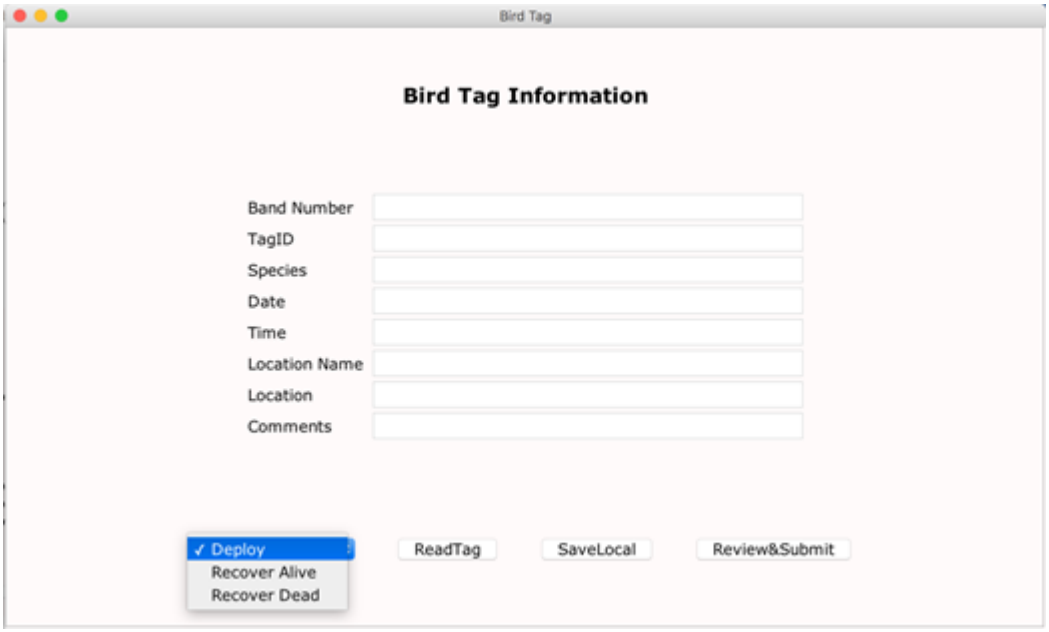


Figure 3.3 Choose Bird Status

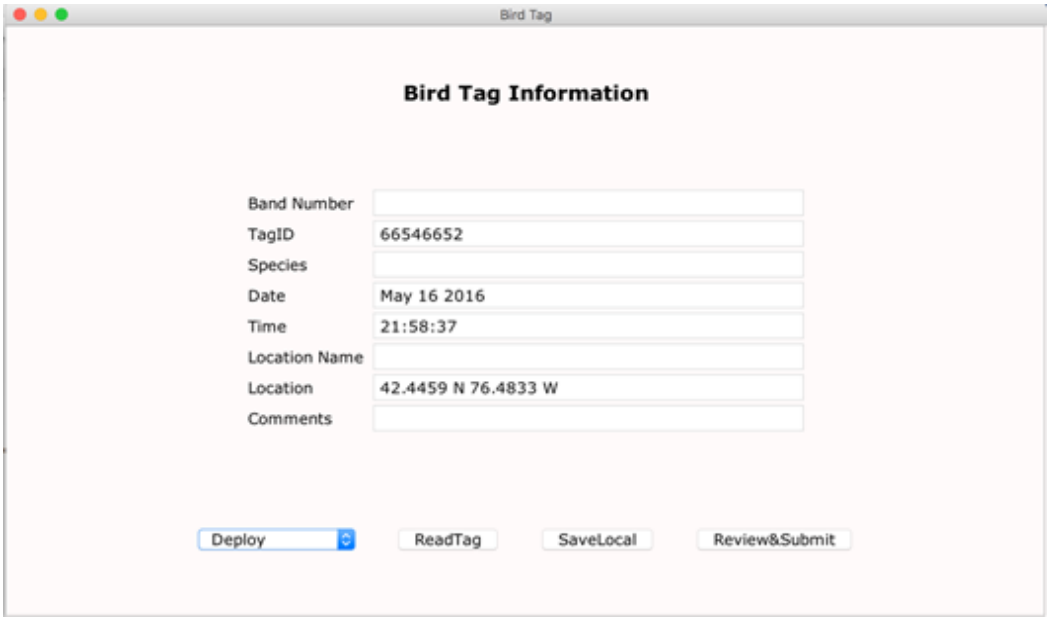


Figure 3.4 After Click ReadTag Button

The third interface shows all the information saved by this user in local computer and user can edit the information and click submit button. At this time, the program would check whether there is internet connection, if it is, program would send JSON file for this information to server and add this information to permanent local txt file, delete the temporary txt file. If there is no internet connection, the file would also save to the temporary txt file.

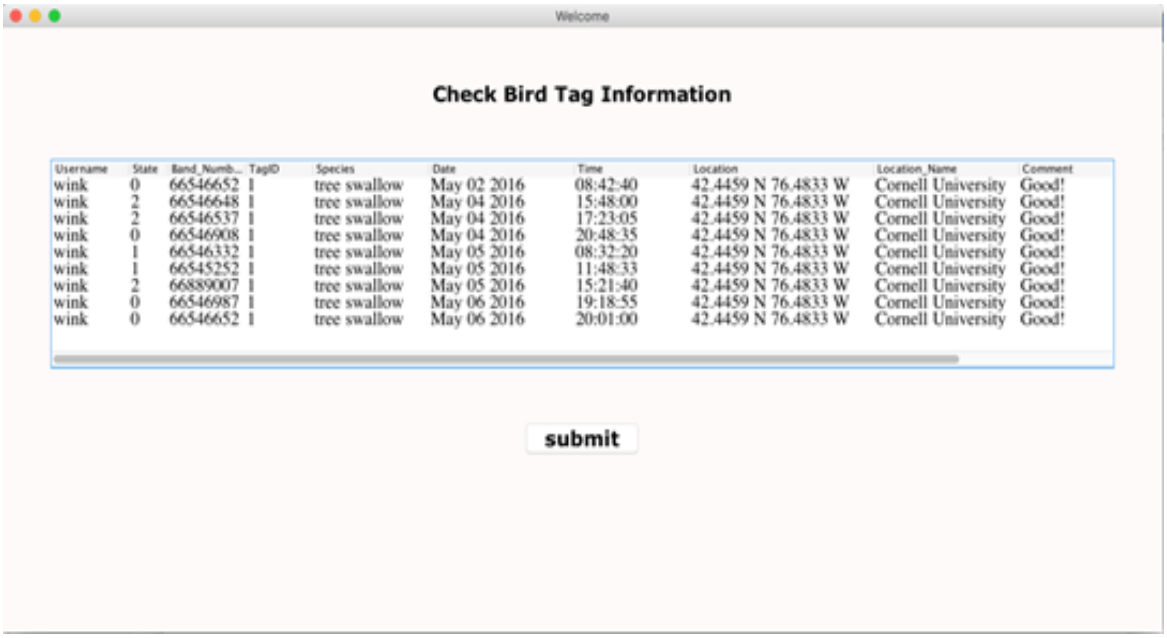


Figure 3.5 Checking Interface

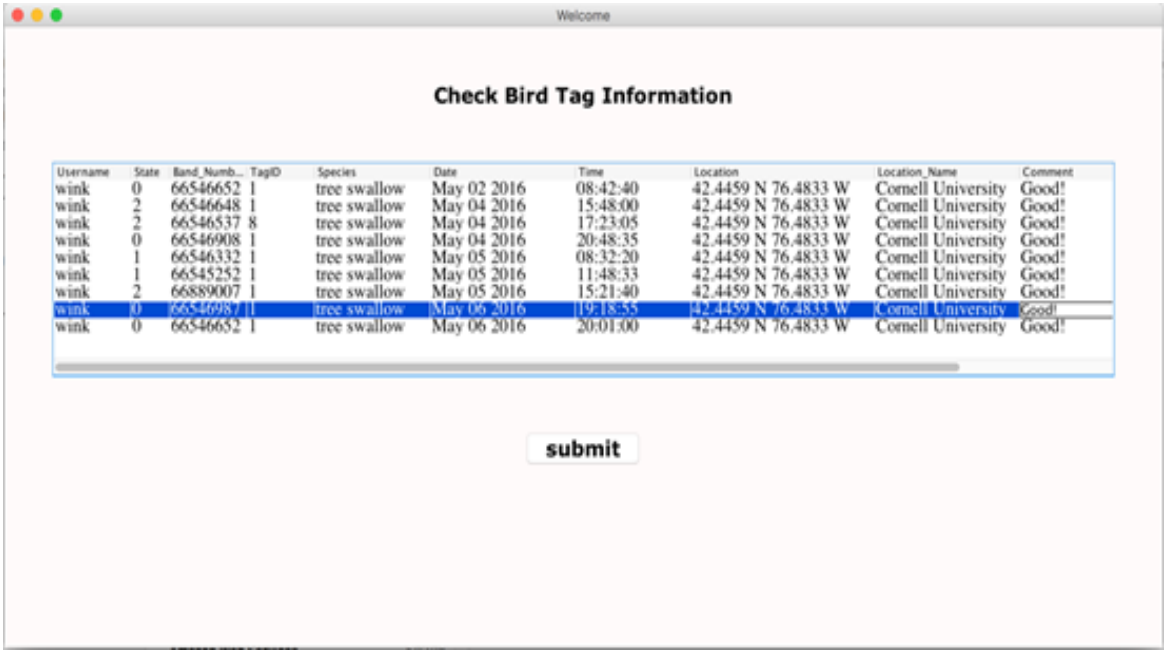


Figure 3.6 Edited Information

3.2.2 Information Receiving

The tagID and GPS information are received through two different USB ports. By using Realterm, the tagID can be watched through baud 115200Hz and the GPS information can be watched through 9600Hz. For GUI interface, we write two protocol for tag ID and GPS information respectively. They work to open the port, read information

periodically, and use regular expression to get useful information and show on the GUI interface.

For Date and Time information, we just use java Calendar API to get the current date and time information.

3.2.3 Local Information Storing

We use txt file to achieve the local computer management function. When user click SaveLocal, all the information will be added to a temporary txt file in computer; in Review part the information from specific user will be shown on the checking interface waiting to be edited and sent to the cloud database. If the information has been sent to cloud database, this information will be deleted form the temporary txt file and stored in local permanent txt file.

3.2.4 Cloud Database Connection

When user click Review&Submit button, finish editing the information and click submit button, the program will ping server address and wait for the feedback to check internet connection, sending JSON file or save information to temporary txt file according to whether the internet is available.

The flowchart of GUI’s working progress is:

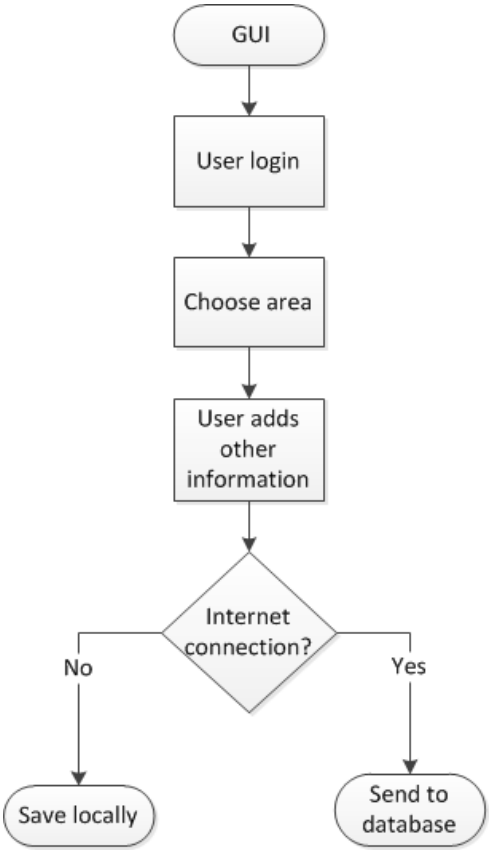


Figure 3.7 GUI's working progress

3.3 Implementation Method

3.3.1 GUI Implementation

The key point in this part is using JAVA AWT and JAVA Swing to implement a user-friendly GUI program including functions, component, layout, and details such as font, size, color and so on.

In the login interface, we use Label as the interface title, JLabel to indicate the different text field, TextField to enter user name and password for user, a button for login and a JComboBox to implement dropdown menu. When user clicks the login button, the program would send the user name and location information from dropdown menu to the information receiving interface. We use FlowLayout for this interface, the labels and Text Field for user name and password use GridBagLayout, and we put dropdown menu and login button to a panel using FlowLayout.

In Information Receiving interface:

1. We add eight JLabels and eight TextFields for different information, and use GridBagLayout to arrange the position for each information.
2. We create a FlowLayout JPanel for three button and a dropdown menu. The dropdown menu is JComboBox, which can record the status of bird, it has three options, Deploy, Recover Alive and Recover Dead. This status will be stored with other information in txt file. ReadTag button uses Button to implement, which can call function to read TagID, GPS information, current Date, current Time and fill in the corresponding field.
3. Other TextFields such as Band Number, Species, Location Name and Comments are also editable and user can type any information they want with the accepted characters, numbers and punctuations.
4. SaveLocal and Review&Submit are also Buttons, which call different methods will be described later to achieve the functions.

The Check Bird Tag Information using JTable and a button to check and edit information before submit. JTable is a component which is easy to edit information. Title, JTable and the Button using FlowLayout to add on the interface.

3.3.2 Data Receiving

The overall sequence of data flow in this system is described in figure 3.9. And the data receiving function only deal with what the GUI do with the received data.

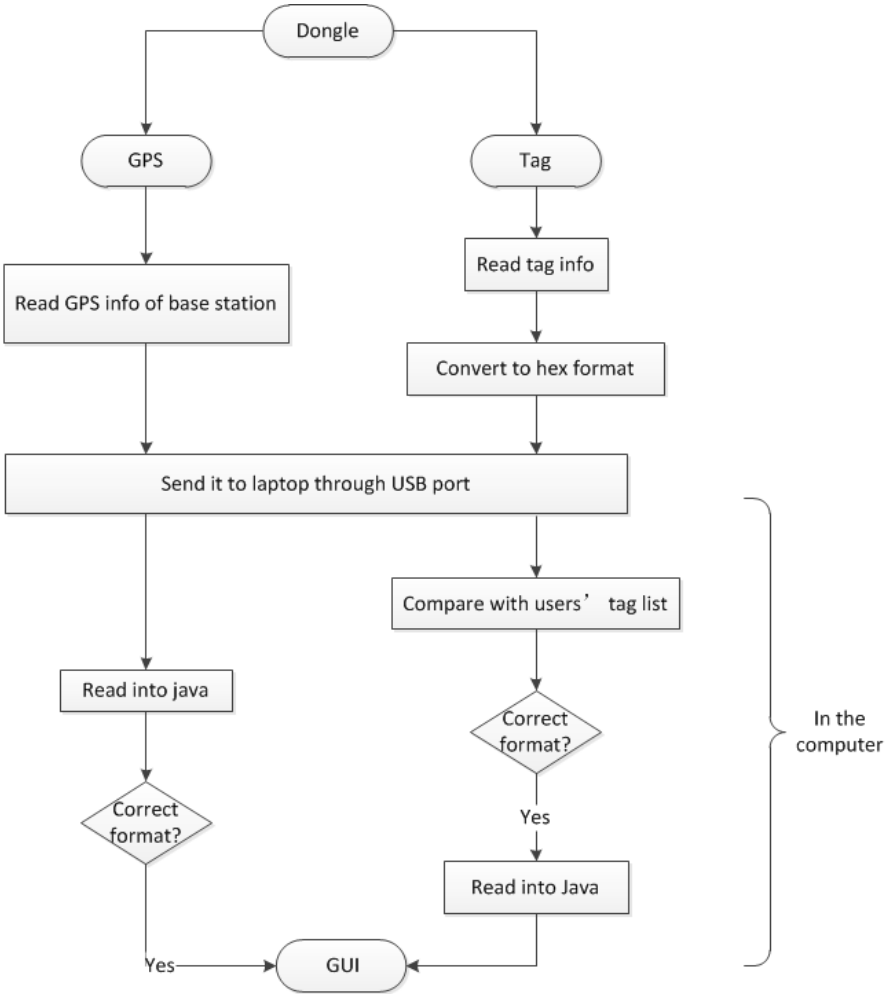


Figure 3.8 Data flow in this system.

This function can receive both TagID and GPS information. For TagID, when a bird fly by Dongle, the Dongle can catch the TagID. After user plug Dongle in the computer, when user clicks readTag button, program will open the ports which Dongle and GPS receiver plugged in and read information from them constantly.

1. GPS information:

The GPS receiver's output stream contains information that the GUI needs, such as the latitude and longitude of the fixed location. But when the receiver is turned on, it takes some time to get the first fix. Before that, the information it transmits is meaningless for the GUI user. The useful information starts with "$GPGGA" or "$GPGLL" and strings like these. So we need to pick out these messages. Meantime, for that different messages contains time and GPS coordinates repeatedly, we only need to pick out the GPGGA message. The format check is through regular expression and implemented by

Pattern and Matcher class in JAVA. If it is right, we extracted latitude and longitude and then transmit them to GUI to display. The format of $GPGGA is as follows:

Structure:

$GPGGA,hhmmss.sss,ddmm.mmmm,a,dddmm.mmmm,a,x,xx,x.x,x.x,M,,,,xxxx*hh<CR><LF>

1 2 3 4 5 6 7 8 9 10 11

Example:

$GPGGA,111636.932,2447.0949,N,12100.5223,E,1,11,0.8,118.2,M,,,,0000*02<CR><LF>

Figure 3.9 GPGGA format

So we wrote the regular expression according to the right format:
RegEx for $GPGGA= ($GPGGA),+[0-2]+[0-3]+[0-5]+d{1}+[0-5]+d{1}.+d{3}+d{4}.+d{4},+(N|S),+d{5}.+d{4},+(E|W),+[0-8],+(0|1)+[0-2]+d{1,2}.+d{1},+(M),+(0.0),+(M),,(0000)+*+d{1}+(d{1}|[A-F])

After that, program will also get current Date and Time, transfer them to String and shown on the GUI. RXTXcomm.jar is the key library we used in this part.

2. tagID information

When program reads a tagID from the port, it will send this information to RegexTextTag.java to check whether this tagID is in valid format (8 hex number).The check is also through regular expression. If the format is right, we will use a string to store it and then transmit it to the GUI. It will then be shown on the corresponding TextField. When the tag is changed, the relevant information will change accordingly. The information can also be checked using Realterm.

3.3.3 Local Information Storing

When user click SaveLocal button in the second interface, program should start to process the saving function. Our goal in this part is check all information once to make sure there is no invalid character and save it in local temporary txt file. We use regular expression to check information and we set different standard for each field. For example, TagID, we parse region code from login interface and call specific TagID for this region and check the format; comment part, we set a to z, A to Z, 0 to 9, some basic punctuations as valid character like below:

String regEx = "^[0-9a-zA-Z:,_!?.;]{1,200}$";
If there is invalid character, program will record the label and pop up an alert box to indicate which field is invalid.

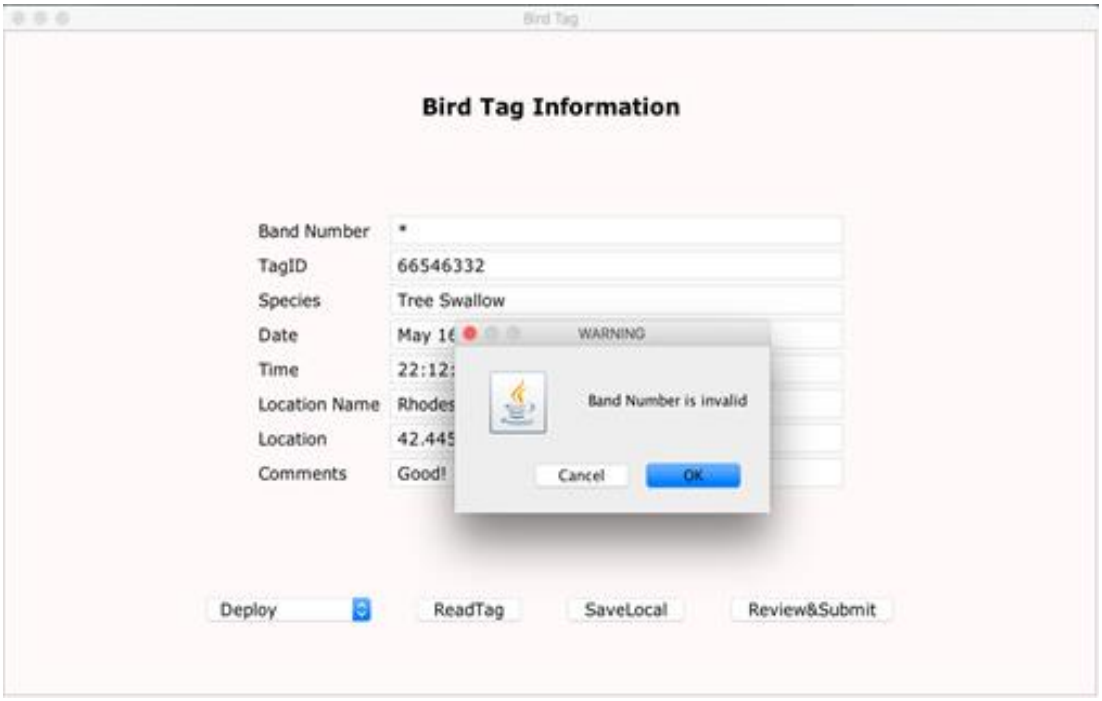


Figure 3.10 Alert Box for Invalid information

Only all information is in correct format can program open the temporary txt file, grab the username, bird status and information shown on GUI and save them to that txt file. What's more, the TextField will be cleaned up when information saves to txt file successfully to prepare to another transaction.

3.3.4 Cloud Database Connection

This function will be called when user clicks Review&Submit button aims to check information again, check internet connection and send JSON file to cloud database. We create this function because biologists want to check the information again to make sure the correction about database. Program will read the temporary txt file and grab the information which type by this user and put them on the interface. All the table cells are editable so that user can change the information if there is typo or invalid information. After reviewing, user could click the submit button. In this part, we use 'ping' command to check whether the internet is connected between client side and server. Program will call CreateJson.java to form a JSON file include all the reviewed information and send it to cloud database. We

consider this information is correct so that we will delete it from temporary txt file and add them to local permanent txt file as a local database.

3.4 Failures and Solution

We faced some difficulties in process of coding, some came from difference between window and mac operating system, some came from the complexity of the tasks and we will describe them in detail.

3.4.1 JPassword TextField

Java provide a component named JPassword TextField which is easy to achieve the password function, however when we run the program JPassword TextField and normal TextField have different size in mac computer and looks normal in windows computer. We reset the size, dimension and tried several times, all the methods are not useful. After that, we have to change back to normal TextField, and use setEchoChar method to set all character typed into this field as '*' to protect account security.

3.4.2 Check Internet Connection

When we try to check internet connection, we use ping command to implement. As the property in windows computer, it will send four packets and there will be four successful feedbacks and stop after that, so we read every feedback line and judge the connection by checking whether the ttl number and TTL number are greater than 0. If yes, it means we ping successfully and we can start to create JSON file. However, when we run this program in mac computer, it is not working because the feedback will not stop in ping command so it will never go out of the reading loop. So we add a counter to count the number of feedback in the loop of reading ping feedback lines, when the number achieves to 3 (index starts from 0), though the ping doesn't stop, it will jump out of the reading feedback loop and go to check the TTL number.

3.4.3 Txt File Management

This function used to achieve save, delete both temporary and permanent txt file. Save local is kind of easy as long as we set FileWriter (filename, true) so that we can open the file if it exists instead of create and cover the temporary file again and again. When

we try to review the information for particular user, we have to read information saved by this user which is easy to parse the username from txt file. And after editing by user, we should either delete this information from txt file and save it to permanent file or save the edited version information to cover the previous version again. These processes sound complex and do complex so we use a temporary inner file to achieve.

When user wants review the information, program will grab information from temporary file, split from long String to different part and store in a two-dimension string array, this information deleted from temporary file at this time. After sending JSON file, the temporary file has already deleted this information, and we can refresh the two-dimension array with edited information, form string and add them to permanent txt file. If we cannot send out the JSON file, we also refresh the array, and save this information to temporary file again.

4. Tag

4.1 Objective

Tags on birds are essential parts of this system. Our base stations and dongles need to communicate with the tags to know where the birds are. Thus enough power must be provided at any given time and the RF signals must be strong enough to be heard by base stations in a relatively far distance. For now we are using tags on tree swallows that average 13.5 cm (5.3 in) long and weighs about 20 g (0.71 oz), so our tag's size must fit into the bird's size: it cannot be so big or heavy that it affects the normal life of the bird. Besides, the biologists want the tag to have a wider range of operating frequencies so they can use it for the current 434 MHz design and also for tags and base stations that will operate at 166 MHz for the MOTUS Wildlife Tracking System.

4.2 Detailed design

4.2.1 Selection of RF chip

There are a lot of RF chips in market. However, in order to select the most suitable chip, we made a lot of surveys and discussed with our

advisor. We considered the price, the size, the weight, and the distance limit the chip can emit signal, and the frequencies it can listen to as well. And after sincere consideration, we chose the si1060 RF chip finally. Because besides the appropriate price, size, and weight, the si1060 chip has many advantages. This chip can not only transmit signals, it can also receive signals as well. In this way, we can use the same chip on RF Dongle. The chip can also support a wide range of operating frequencies so we can use it for the current 434 MHz design and also for tags and base stations that will operate at 166 MHz for the MOTUS Wildlife Tracking System.

4.2.2 The schematic

There are RF chip, solar chip and antenna need to be connected in the small tag. Besides, in order to test the tag through computer, we also add a TC2030-MCP-NL adapter for tag. So connecting all of the components together and make the whole tag work well is of great importance. We discussed together and then designed the schematic like this:

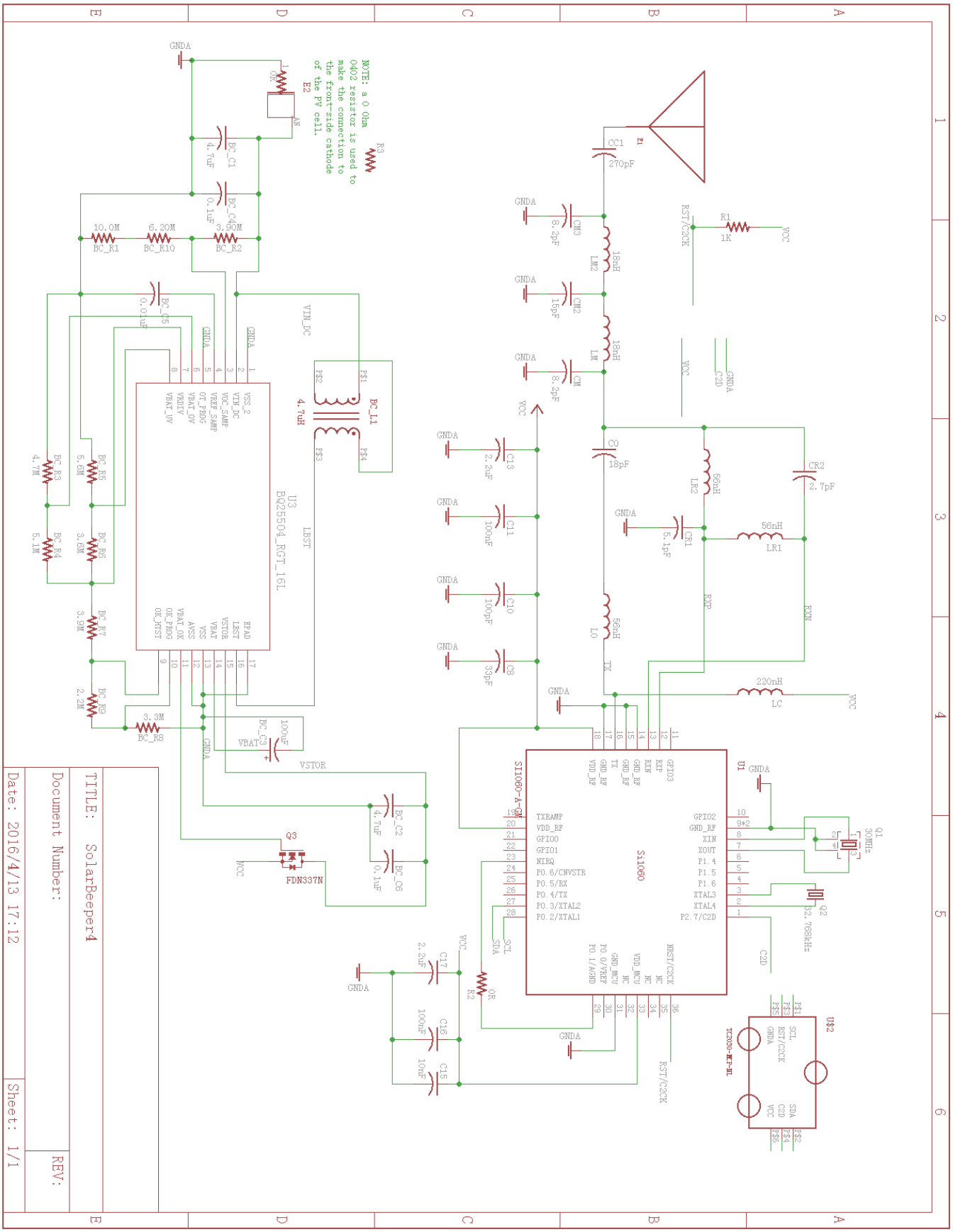


Figure 4.1 Schematic for the new tag

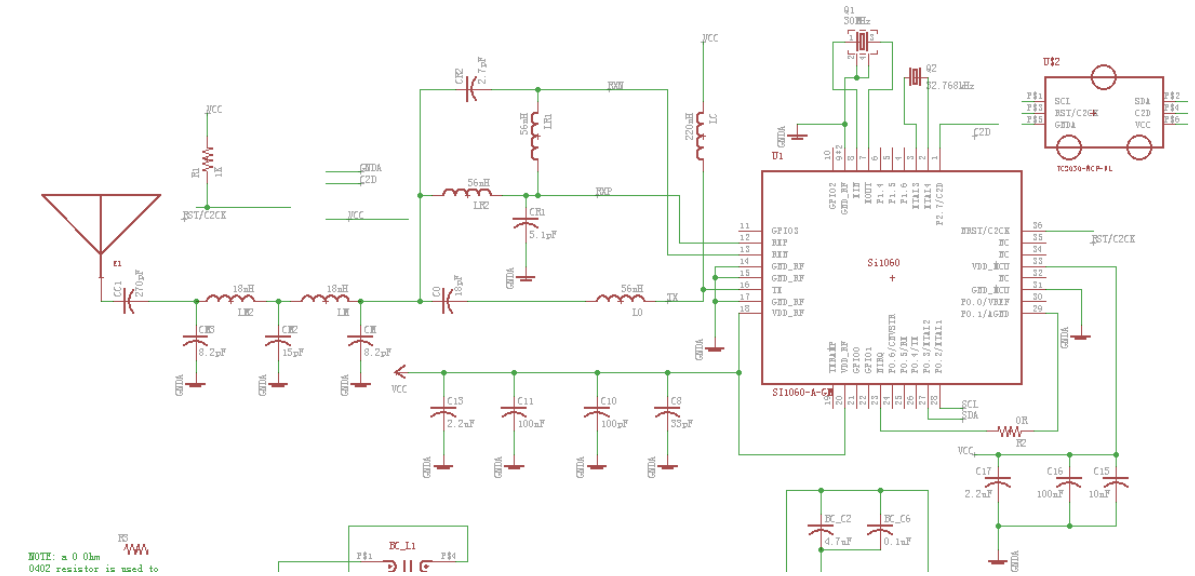


Figure 4.2 Schematic for Si1060 and adapter

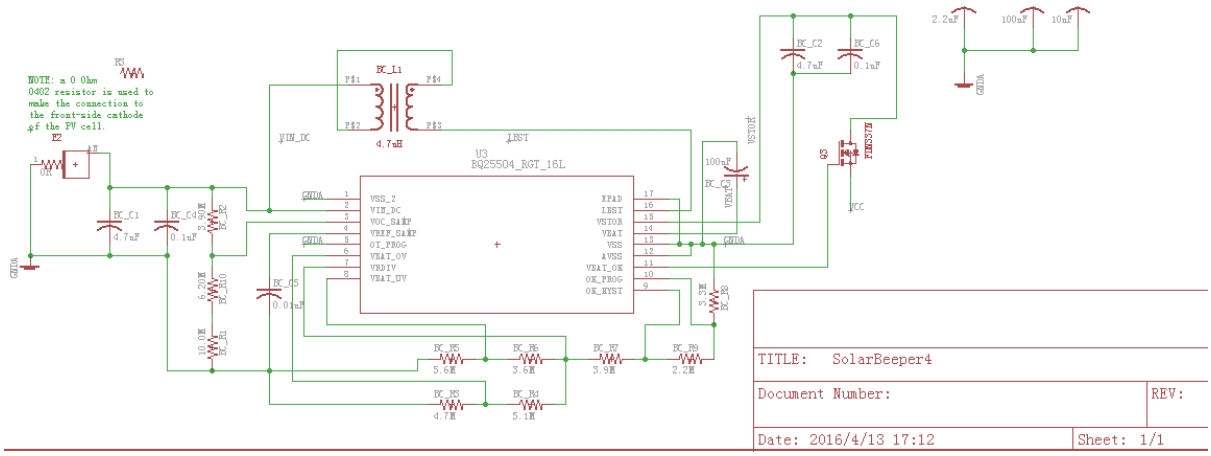


Figure 4.3 Schematic for Solar chip

In order to allow the RF chip to utilize solar energy more efficiently, we designed a volts changing schematic and in this way, the voltage provided for the RF chip became 5V from 3V.

4.2.3 The PCB layout

The tag is designed to be really small and there are several components of the tag. So how to deploy the lay out appropriately is a hard problem. In order to make good use of the limited space. We decided to use the double-layer layout.

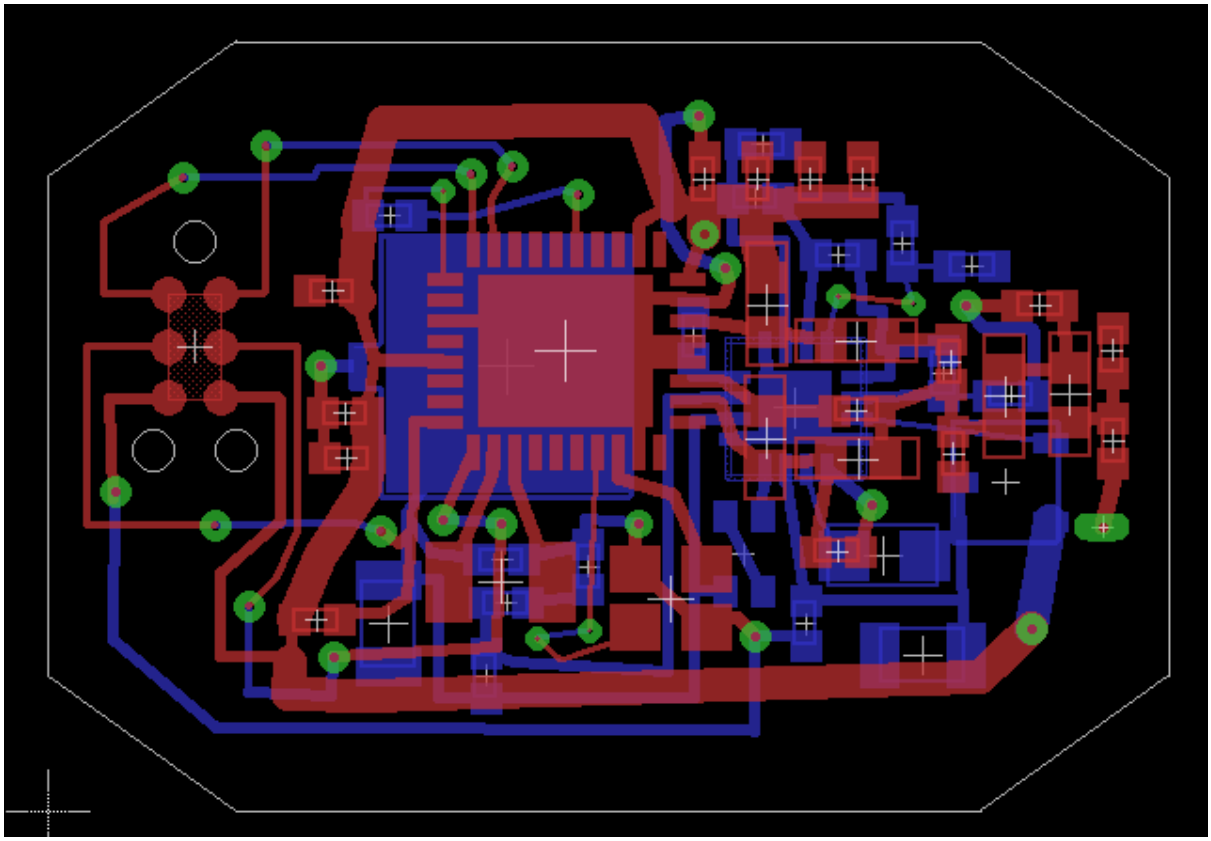


Figure 4.4 PCB double-layer layout

We put the RF chip, and the antenna on the top layer, and the solar chip was placed on the bottom layer. So although the tag is small, we can still arrange the chips well. However, since the schematic is a little complicated, and there are so many wires, most of them crossed together and the lay out was totally a mess at the beginning. After heated discuss and detailed consideration, we adjusted the components layout and altered some wire rooting. For example, we made some wires detour to avoid crossing collision. As for the two spins needed to be connected together but were far away from each other, if both pf them needed to be connected to the ground (GND) too, we changed the wire connection and just made them be connected to GND. And in this way the components in the same layer were arranged pretty well.

As I mentioned before, there are two layers of the tag. And the components not in the same layer need to be connected together too. So we needed the wires cross the layer through the via. And the via cannot be too close to any component, so in order to place the vias, we changed the wires' arrangements again and again, we saw the

errors of the layout decrease step by step from 361 to 0! (We are using Eagle to check board layout.)

After that, in order to make the testing of the tag easy to implement, we decided to add a TC2030-MCP-NL adapter to connect the tag to computer. However, there was hardly any space in the tag. In order to add this adapter, we used “Move Group” function and moved the original layout a little and also changed some wire connections so that we could make some space for the adapter.

Finally, we accomplished this small but efficient tag, and then sent the Eagle file to factory to manufacture the tag.

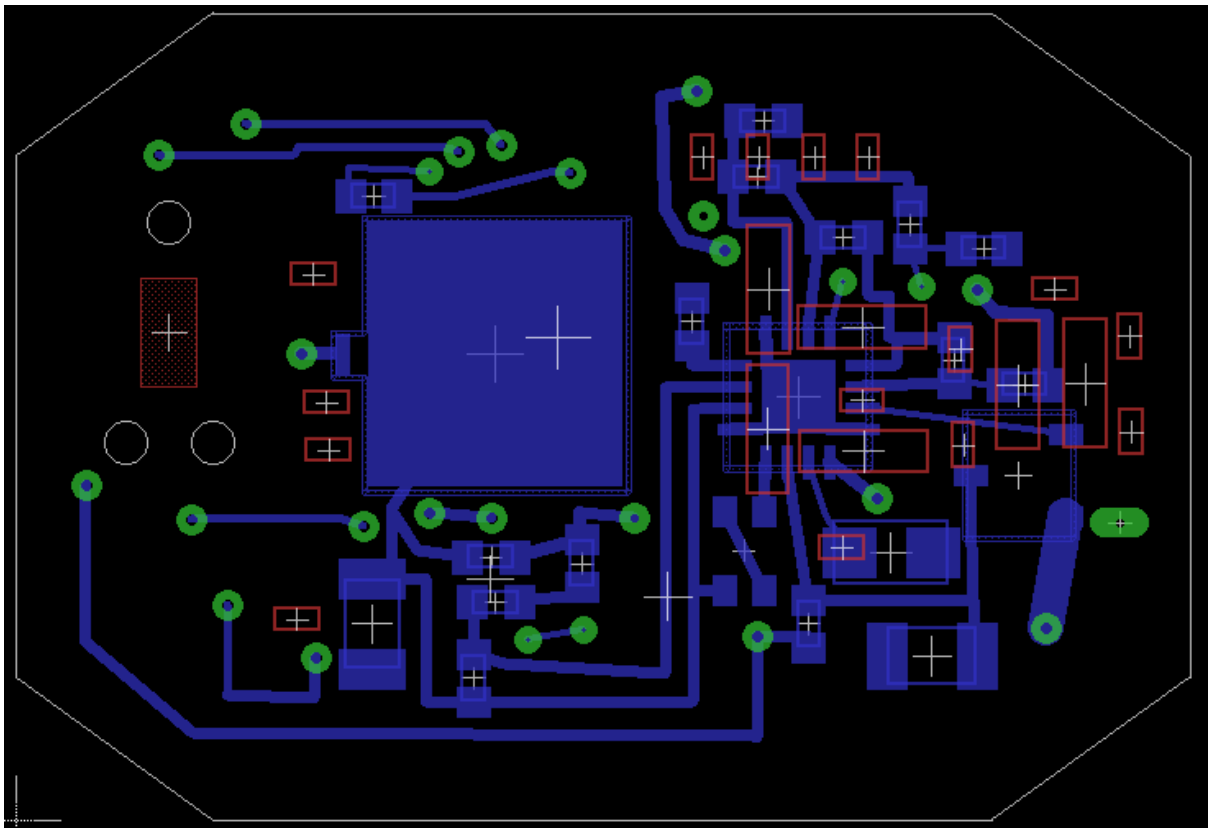


Figure 4.4 PCB bottom layer includes the solar chip

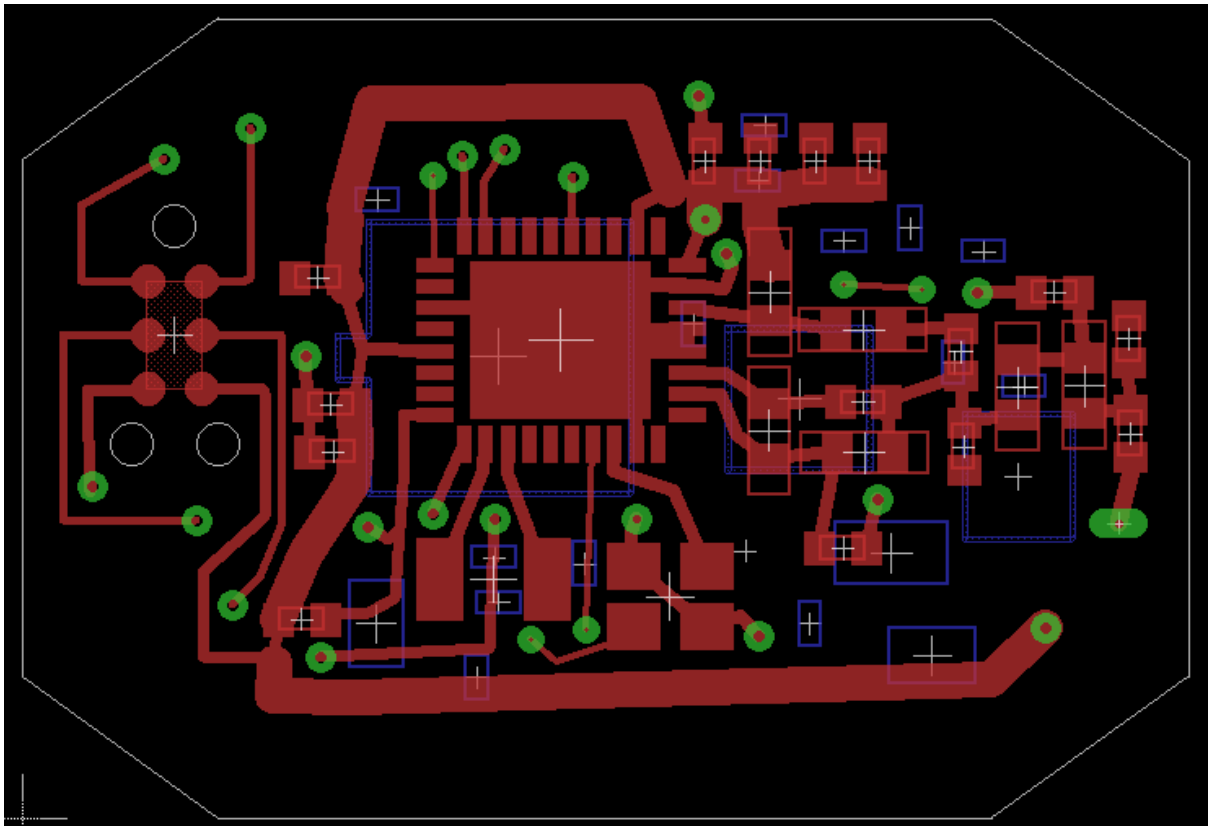


Figure 4.5 PCB top layer includes Si1060, antenna, and the TC2030-MCP-NL adapter.

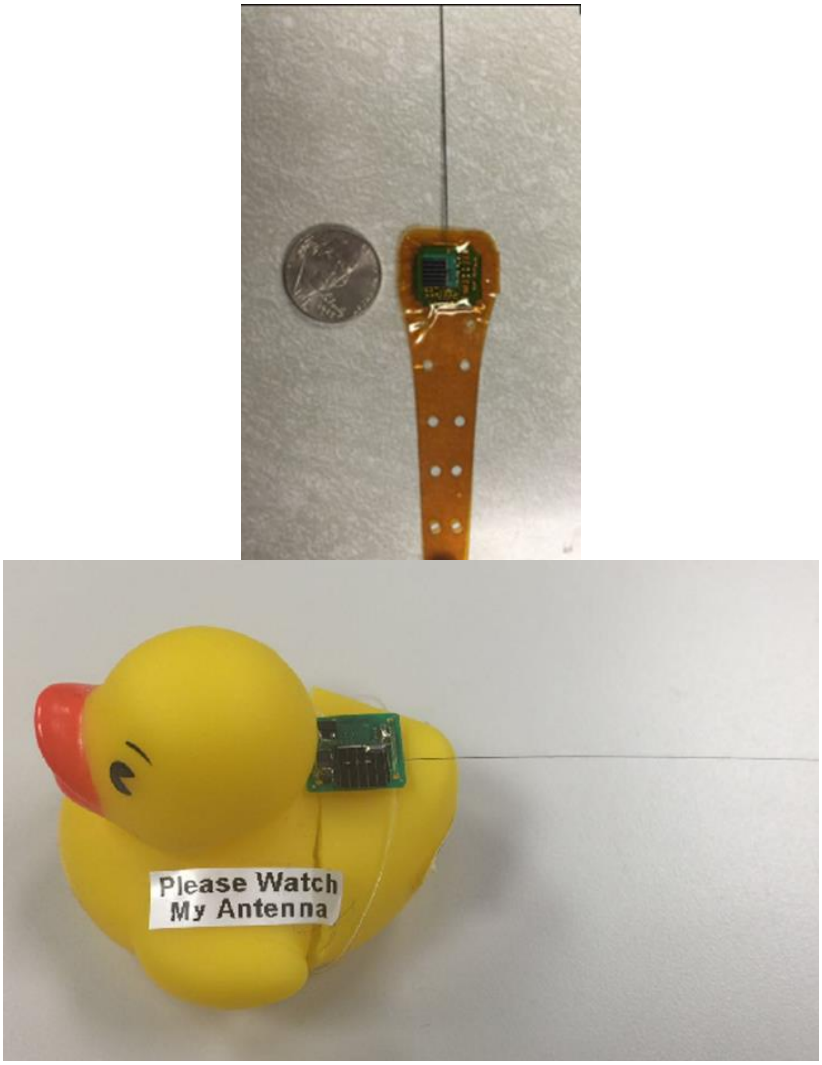


Figure 4.7 a manufactured tag

4.3 Test Result

1. The tag is really small, light and flexible. So it can be attached to birds easily without hurting them.
2. The tag can work normally if there is light, which is really convenient to charge with sunlight.
3. The tag can send signal to Dongle or base station within a mile, so the biologists can manage their birds more conveniently and accurately.

5. Conclusion

1. We programmed on Si1060 to let it listen to certain radio frequency constantly.
2. We configured the Si1060 and USB port adapter correctly so that they could work coordinately.
3. We redesigned the dongle board by combining the 2 parts together in a smaller board in Eagle. We had several samples and tested them. After some adjustments, the dongle we have now is small and highly integrated. Users could use it much more easily.
4. We created a GUI to receive input from the dongle. We applied Regular Expression to check the format of tag ID before reading the information into the GUI, only the tag ID of right format can be read into GUI. Besides, the formats of other fields in the GUI can also be checked. The GUI is adaptive to both Windows and Mac OS.
5. Users could choose areas in login interface, after logging in, users could view and edit tag data before submitting to the database or saving locally, depending on the Internet condition.
6. We redesigned the tag board so that the new tag could listen to a wider range of frequencies and has a higher efficiency on utilizing solar energy, without changing too much on the tag size and tag weight.

6. References

1. Silicon Labs. Si106x/8x data sheet. Retrieved from https://www.silabs.com/Support%20Documents/TechnicalDocs/Si106x-8x.pdf
2. FTDI chip. FT232R USB UART IC Datasheet. Retrieved from http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf
3. EAGLE Tutorial. Retrieved from http://www.cadsoftusa.com/fileadmin/journalist/Documents/V7.3_tutorial_en.pdf

Appendix1 Code

Github source:
https://github.com/CU-TABER/Si1060_GPT_demo
https://github.com/CU-TABER/GUI-Database-Connection-Java-Code

Appendix2 EAGLE Files

Github source:
https://github.com/JinghanDu/TABER