Ningdong Wang (nw3, ECE110)
Andrew Samboy (asamboy2, ECE120)
Moriah Gau (mgau2, ECE110)

Indoor Positioning System

Introduction

Problem Description

The goal of our project was to construct a device that could use RFID sensors to scan an area and locate different objects, in addition to lighting up different color LEDs in respect to the objects that were detected. This task was accomplished through the usage of a XBee, RFID Module, RFID Evaluation Shield, and RFID tags that were soldered together and attached to the Arduino Board, LED circuits, and alkaline batteries.

Design Concept

The main components of our design are the XBee, RFID Module, RFID Evaluation Shield, and the RFID tags. The XBee, RFID Module, and RFID Evaluation Shield were soldered and mounted on top of one another and the inputs of the RFID Evaluation Shield were all connected to the inputs of the Arduino board. The RFID Evaluation Shield would detect the presence of the RFID tags if they are within its detection parameters and would send a signal input to the RFID Module which would send another individualized input to the Arduino board in relation to the ID code of the RFID tag. Based on the the input received from the RFID Module, the Arduino will then run through the code and convert the ID of the RFID tag to HEX and display it on the screen and the corresponding colored LEDs would light up.

Analysis of Components

RFID Module - SM130 MIFARE

The SM130 is a 28 pin DIP module that includes all necessary components for a 13.56 MHz RFID MIFARE®. But it doesn't include an antenna, so we need an additional antenna for this module to work. It communicates over UART or I2C, but the module firmware needs to be upgraded for I2C operation, so we used UART to control the module. The module has 23 simple command protocols to interact with RFID tags. It also has 2 general purpose inputs and 2 general purpose outputs.

RFID Evaluation Shield

This evaluation board includes PCB antenna, an XBee header and can be used as a shield for Arduino. It occupies digital pin 7 and 8 as TX and RX to communicate with the module and pin 11 and 12 to control two LEDs “error” and “found”. It can be easily mounted on top of Arduino and hold the XBee and RFID module.

The antenna can detect MIFARE Classic 1K tags 5’’ above with no obstacle in between. The reader have difficulty detecting tags attached to electronic devices like phones and laptops.

RFID Tag

This RFID tag functions within the MIFARE Classic 1K guidelines. It has 1K of data storage and an adhesive backing. The tag has a diameter of 30mm with an overall thickness of 0.4mm.

XBee Module

The evaluation board has an XBee header that works with the XBee Explorer Regulated board. Although wireless communication is not used in our project at the moment, this feature can be easily included with some more coding. It will allow users to send tag serial numbers remotely to computers.

Code:

```
/*
  RFID Eval 13.56MHz Shield example sketch v10

  Aaron Weiss, aaron at sparkfun dot com
  OSHW license: http://freedomdefined.org/OSHW

  works with 13.56MHz MiFare 1k tags

  Based on hardware v13:
  D7 -> RFID RX
  D8 -> RFID TX
  D9 -> XBee TX
  D10 -> XBee RX

  Note: RFID Reset attached to D13 (aka status LED)

  Note: be sure include the SoftwareSerial lib, http://arduiniana.org/libraries/newsoftserial/
```

```
  Usage: Sketch prints 'Start' and waits for a tag. When a tag is in range, the shield reads the tag,
  blinks the 'Found' LED and prints the serial number of the tag to the serial port
  and the XBee port.

  06/04/2013 - Modified for compatibility with Arudino 1.0. Seb Madgwick.

*/
#include <SoftwareSerial.h>

SoftwareSerial rfid(7, 8);
SoftwareSerial xbee(10, 9);

//Prototypes
void check_for_notag(void);
void halt(void);
void parse(void);
void print_serial(void);
void read_serial(void);
void seek(void);
void set_flag(void);

//Global var
int flag = 0;
int Str1[11];

//INIT
void setup()
{
  Serial.begin(9600);
  Serial.println("Start");

  // set the data rate for the SoftwareSerial ports
  xbee.begin(9600);
  rfid.begin(19200);
  delay(10);
  halt();
}

//MAIN
void loop()
{
  read_serial();
}

void check_for_notag()
{
  seek();
```

```
  delay(10);
  parse();
  set_flag();

  if(flag = 1){
    seek();
    delay(10);
    parse();
  }
}

void halt()
{
 //Halt tag
  rfid.write((uint8_t)255);
  rfid.write((uint8_t)0);
  rfid.write((uint8_t)1);
  rfid.write((uint8_t)147);
  rfid.write((uint8_t)148);
}

void parse()
{
  while(rfid.available()){
    if(rfid.read() == 255){
      for(int i=1;i<11;i++){
        Str1[i]= rfid.read();
      }
    }
  }
}

void abstract()
{
  //print to serial port
      Serial.print(Str1[8], HEX);
      Serial.print(Str1[7], HEX);
      Serial.print(Str1[6], HEX);
      Serial.print(Str1[5], HEX);
      Serial.println();
      //print to XBee module
      xbee.print(Str1[8], HEX);
      xbee.print(Str1[7], HEX);
      xbee.print(Str1[6], HEX);
      xbee.print(Str1[5], HEX);
      xbee.println();
      delay(100);
      //check_for_notag();
```

```
}

void print_serial()
{
 if(flag == 1){
    if(Str1[8] == 180) {
       if(Str1[7] == 117) {
          if((Str1[6] == 231) && (Str1[5] == 238)) {
                Serial.println("Andy");
                digitalWrite(2,HIGH);
                delay(1000);
                digitalWrite(2,LOW);
          } else if ((Str1[6] == 220) && (Str1[5] == 206)) {
                Serial.println("Ningdong");
                digitalWrite(3,HIGH);
                delay(1000);
                digitalWrite(3,LOW);
          } else if ((Str1[6] == 235) && (Str1[5] == 62)) {
                Serial.println("Moriah");
                digitalWrite(4,HIGH);
                delay(1000);
                digitalWrite(4,LOW);
          } else {
                abstract();
          }
       }else {
         abstract();
       }
    } else {
      abstract();
    }
 }
}

void read_serial()
{
 seek();
 delay(10);
 parse();
 set_flag();
 print_serial();
 delay(100);
}

void seek()
{
 //search for RFID tag
 rfid.write((uint8_t)255);
```

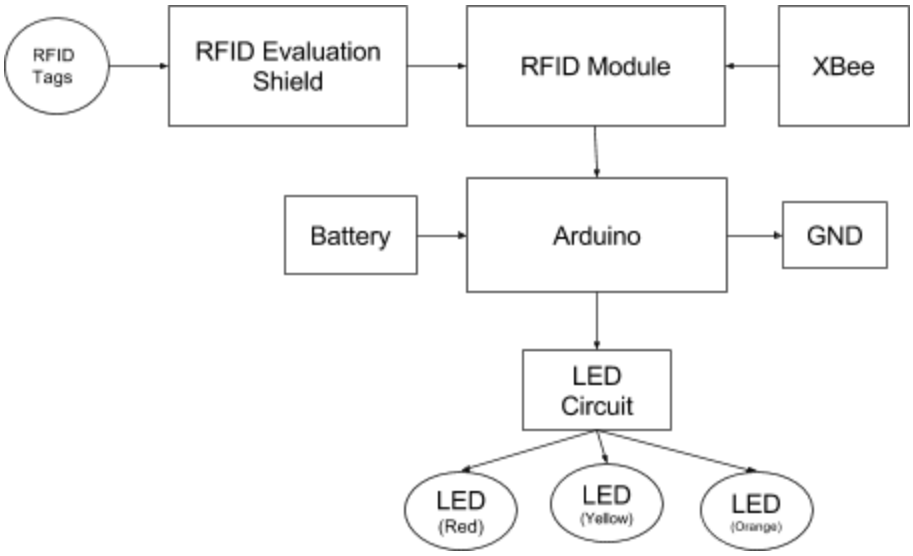
```
  rfid.write((uint8_t)0);
  rfid.write((uint8_t)1);
  rfid.write((uint8_t)130);
  rfid.write((uint8_t)131);
  delay(10);
}

void set_flag()
{
  if(Str1[2] == 6){
    flag++;
  }
  if(Str1[2] == 2){
    flag = 0;
  }
}
```

Design Considerations

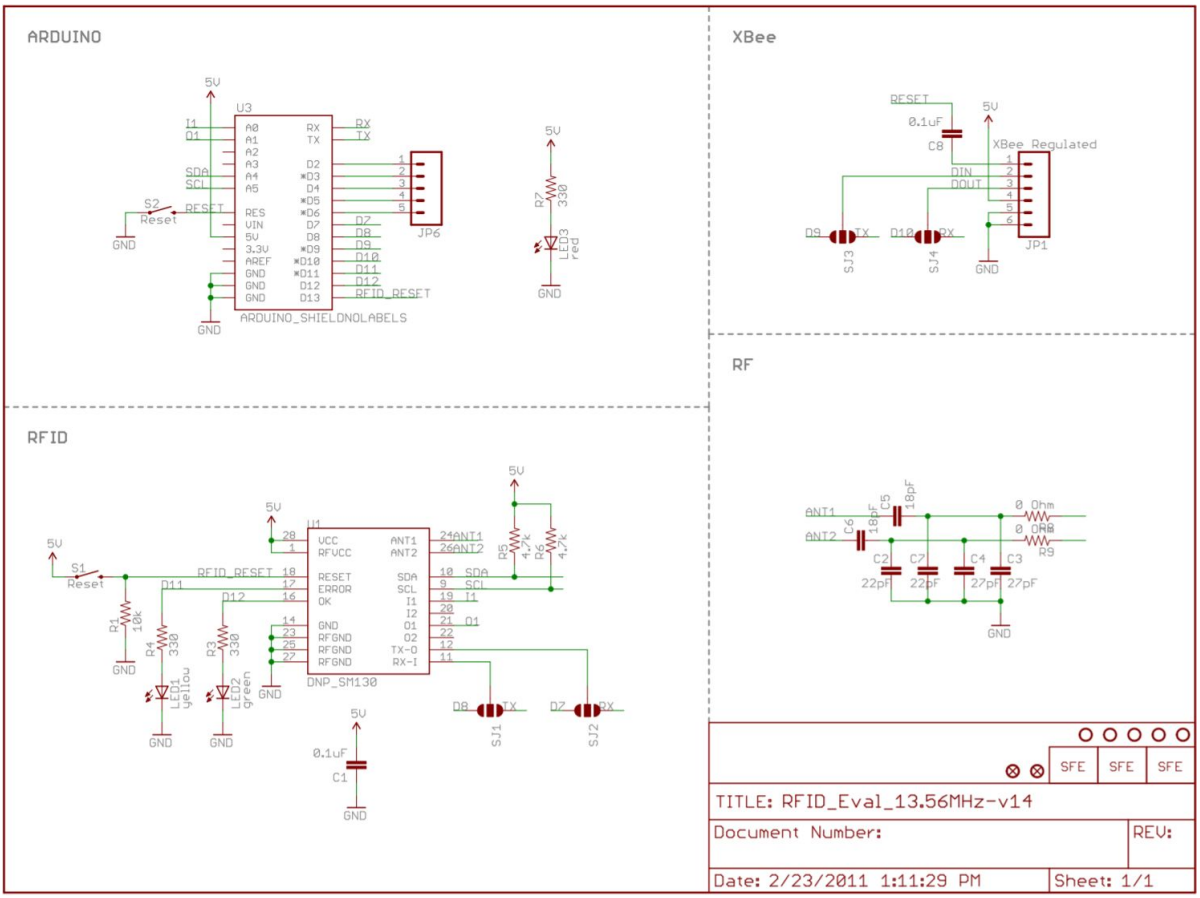
- We want to locate objects remotely while also identify the object. After some research we decided that RFID would be an ideal choice. RFID uses radio frequency to track and identify tags, which satisfies our design need.
- RFID uses several different frequencies to communicate. The SM130 MIFARE uses 13.56 MHz. This frequency is a popular standard HF RFID and is widely used. Its read range varies between 10cm and 1m.
- How many LEDs? The RFID module only had two outputs so we decided to have less than 4 LEDs as to avoid having to add outputs (2bits = 2^2 possibilities = 4 outputs)

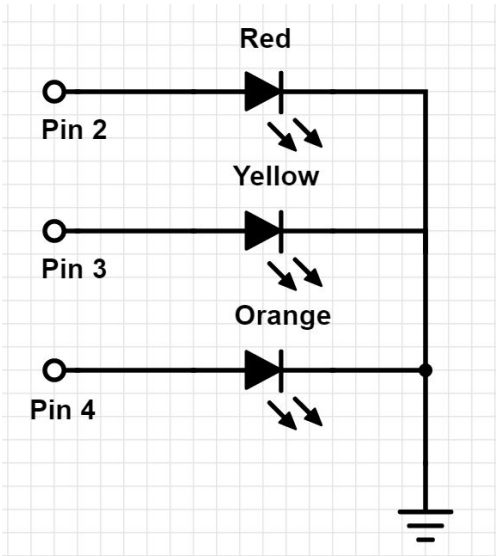
Design Description

Block Diagram



Circuit Schematic





Physical/Mechanical Construction

Our TA helped us to solder the module and the evaluation board together. However, as we didn't have the correct long headers, the board couldn't be mounted directly on the Arduino. We had to use wires to manually connect corresponding pins.

We then built the LED circuit on a breadboard using a common ground and separate input pins.

Conclusion

Lessons Learned

We learned to use RFID technology to read the serial number from tags and convert it into signals to light up different LEDs. We also used simple UART protocols to send command from Arduino to control the RFID module and get responses.

Self-Assessment

Our Indoor Positioning System was able to function efficiently and fulfilled its purpose of detecting items and lighting the correct LED according to the RFID tag ID. However, one difficulty that our group faced was that while the RFID Evaluation Shield and the Arduino Code would most often be accurate and display the correct RFID tag ID, there were a few instances where there were errors and the display of the RFID would show an invalid/unregistered ID. If we were given additional time, we should be able to solve this problem by debugging our code and fixing any errors that could be the cause. However, overall, the Indoor Positioning System was capable of functioning effectively and can be considered successful.