RFID Check-in System

for Safe Routes to School at Hawthorne Elementary

by

Paul Banel

Senior Project

ELECTRICAL ENGINEERING DEPARTMENT
California Polytechnic State University
San Luis Obispo
Spring 2012

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

Acknowledgements

I would like to thank my advisor Dr. Bridget Benson for her support throughout the project. I would also like to thank Dr. Tali Freed for inspiring me to work with RFID technology in my senior project and for putting me in contact with the Safe Routes to School committee at Hawthorne Elementary. I would also like to thank Michael Masuda and Kenny Schmutz for working with me on the preliminary idea for the system during Dr. Freed's RFID design course in the fall. Finally, I would like to thank Rod Hoadley, Tom di Santo, and the rest of the Safe Routes program at Hawthorne for the opportunity to create this project to help fulfill a need in the community and for feedback during the course of the project.

Abstract

The system uses a low frequency Radio Frequency Identification (RFID) reader and passive RFID tags to help the Safe Routes to School program at Hawthorne Elementary in San Luis Obispo track the number of children who participate in the program. Children will carry passive RFID tags and scan them at the reader each morning that they walk or bike to school. The system tracks the total number of students that walk and bike each day, as well as the number of times each individual child participates over the course of many days. The Safe Routes program hopes to use this data to apply for funding from the national Safe Routes to School program in order to improve route safety, promote walking and biking, and offer incentives for participating in the Safe Routes program at Hawthorne. The system successfully tracked students on two separate testing days, and after feedback was revised to include the ability to track students who carpool or bus to school. The Safe Routes program hopes to use the system for data collection starting in the 2012-13 school year.

I. Introduction

In 2009, only 35% of students in grades K-8th who lived within one mile of their school walked or biked to school. This is down from 89% of children living within one mile of school in 1969. Even from just 1995 to 2009, arrival to school by a personal vehicle increased from 45% to 51%. As shown in Figure 1, even of the students living within only one-quarter mile of their school, one third arrived by personal vehicle [3].

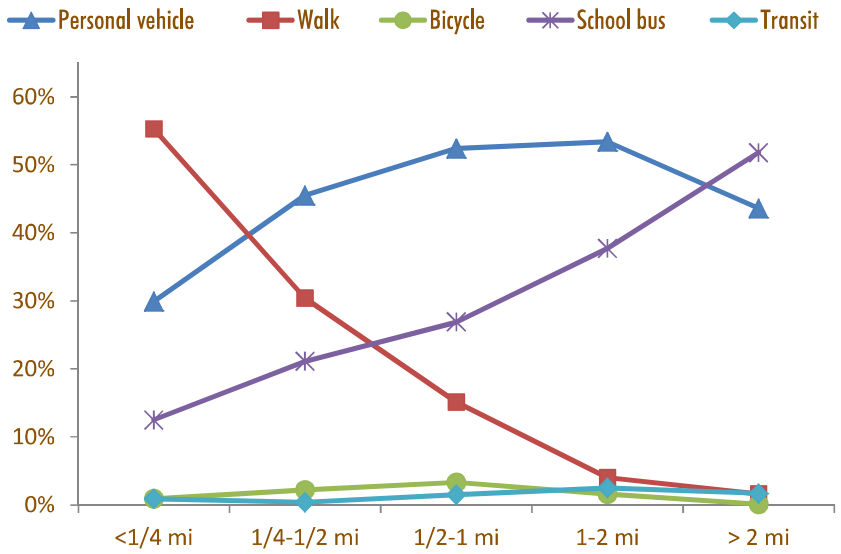


Figure 1: Distance students lived from their school vs. how they arrived, 2009 [3]

Additionally, since 1995, the overall number of students walking, biking, or riding the bus to school from any distance has declined, and the number of students arriving by a family vehicle has increased, as shown in Figure 2 [3].

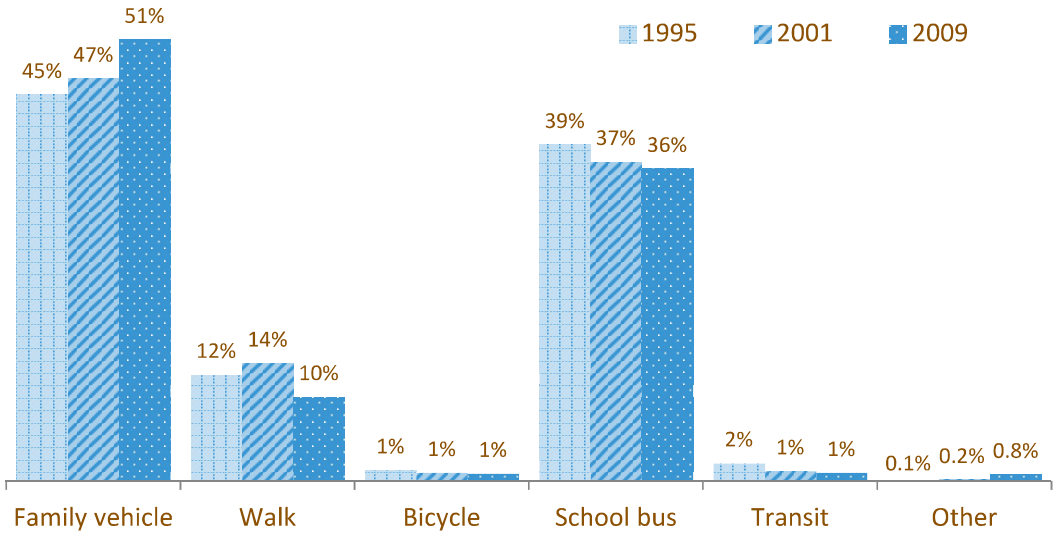


Figure 2: Methods of transportation to school in the U.S., grades K-8th [3]

The Safe Routes to School National Partnership was launched in 2005 with the following mission:

> The Safe Routes to School National Partnership's mission is to advocate for safe walking and bicycling to and from schools, and in daily life, to improve the health and well-being of America's children and to foster the creation of livable, sustainable communities [2].

The Safe Routes program at Hawthorne Elementary also hopes to encourage walking and biking to school and to improve the safety and accessibility of routes to and from school. The National Partnership focuses on the 5 E's: Evaluation, Engineering, Education, Encouragement, and Enforcement [2]. This project specifically focuses on the Evaluation and Encouragement aspects: evaluating trends in how children commute to school through the use of passive Radio Frequency Identification (RFID) tag system and encouraging children to walk and bike by logging how many time each individual student walks or bikes, in order to offer incentives. Using this data collected in the Evaluation step, a local program can apply for grants the national program offers to help improve programs across the country. The Safe Routes program at Hawthorne hopes to collect data during the 2012-13 school year in order to apply for grant money to improve routes, encourage children to participate in the program, and potentially upgrade to an automated RFID system. If successful, the Safe Routes program could share the system design with other local schools for the same purpose.

II. Background

General RFID functioning and uses

Radio Frequency Identification (RFID) systems use wireless communication between a reader and a tag. The reader sends a signal, to which a tag responds with its own signal. All tags contain an antenna and an IC, which stores the data and controls how the tag responds to a reader. A passive tag does not contain a battery and uses the signal sent by the reader to power its response signal through the technique called backscattering. Generally, passive tags have a low read range and low amount of data storage. An active tag contains a battery to power its response signal, and can have much further read ranges. Tags are classified by the amount of type of tag, amount of storage, and functions it can perform [5]. The tags used by the system presented in this report are Class 0 (passive, read-only tags) and each contains only a unique 10-digit hex ID number [4]. RFID systems are used most commonly for inventory tracking. Inexpensive, passive tags are generally used on smaller items such as pallets or even single products. Active tags are used on larger items, such as shipping containers, and can also utilize real-time locating through GPS [5].

RFID as a method for tracking students

RFID systems have also been designed for the purpose of tracking students who bike and walk to school. One of the more elaborate systems, designed by the Boltage organization, uses a solar-powered RFID reader (shown in Figure 3) that connects to a local network via Wi-Fi [1].

Figure 3: The Boltage RFID reader [1]

Students wear a passive tag on their backpack, and the reader automatically scans them when they pass through its range. The reader then passes this scan to the local network and the Internet, where the data is uploaded and stored on Boltage's server. Students can view their participation on Boltage's website and schools can view and use the data to study trends and offer incentives to students who participate. Boltage offers this system at the cost of approximately $5000 [1].

Initial project ideas

Unfortunately, a system such as the one designed by Boltage is beyond the current budget of the Safe Routes program at Hawthorne. However, this offered a unique opportunity to design the system presented in this report, where the ultimate goal was to design a cost-effective

method of collecting data in order to increase funding to the program in the future. In IME570: Advanced RFID and Electronic Manufacturing during the Fall 2011 quarter, preliminary approaches to this system were explored. Due to the scope of the class, only an initial design idea and cost comparison of a two passive RFID systems were created. In the first system, students would have passive tags on their bike helmets and would utilize an RFID reader portal through which students enter at Hawthorne's bike area (see Figure 4).

Figure 4: Bike area at Hawthorne Elementary

The second system, which was chosen as the basis for this project, used a short-range reader and which students check in by putting their passive tags into the field of the reader, also located at the bike gate. The costs of the systems were estimated at $3000 and $300, respectively. Aside from budget constraints, the second system allows for tracking of all students, whereas the first system discussed requires the tag to be on a student's bike helmet. In order for the first system to read tags in other locations (a backpack, for example), it requires additional reader antennas, further increasing the cost. In this project's proposal, the original estimated cost came to nearly $800 due to the expectation of needed to connect to a wireless network, track many more students than actually required, and a permanent outdoor structure. However, the final cost ended up being lower than the original $300 estimate because these features were not needed.

III. Requirements

The system must record the total number of students who walk, bike, carpool, or bus to school each day, and distinguish between students who walk or bike and students who carpool or bus. Since it will be outside, the system should be in a weatherproof container. The cost of the system should stay as low as possible and be easy to replicate for use at other Safe Routes programs in San Luis Obispo. The system should be user-friendly and a user manual should cover all aspects of use including set-up, operation, and troubleshooting. The user should be able to change, add, or remove RFID tag numbers from the list of known tags that the system reads from in order to properly log each student. Students should be able to easily interact with the system so that accurate log data can be collected. Student volunteers should also be able to set up the system on days when Hawthorne faculty members are not available to do so.

IV. Design

Hardware components

The hardware block diagram is shown in Figure 5. The system uses an Arduino Uno SMD microcontroller (Appendix D, Figure 12) to receive and process data from a serial Parallax RFID Reader (Appendix D, Figure 14). The microcontroller then communicates with the Arduino Wireless SD Shield (Appendix D, Figure 13) to write to a microSD card and the LEDs (Appendix D, Figure 15) to confirm communication with the reader. A watertight plastic container holds all the components (Appendix D, Figure 10 and Figure 11). All student users receive a Parallax 54x85mm passive tag (Appendix D, Figure 14), which they use to check in with the system. The Arduino components and Parallax reader operate at $5V_{DC}$, and the reader communicates with the tags at 125kHz [4].

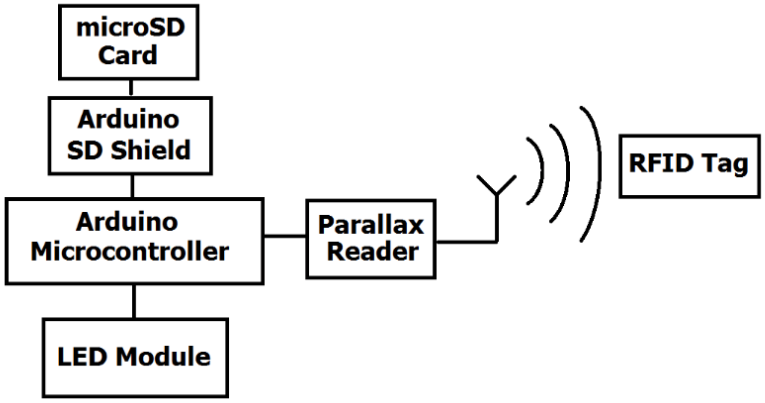


Figure 5: Hardware block diagram

System functioning

Each time the system is powered on a new section of the log begins. For each known and unique tag detected by the reader, the microcontroller writes the associated student ID number to the log file on the microSD card. If the reader detects an unknown tag, the microcontroller writes a generic student ID to the microSD card. In this way, the system differentiates between students who walk and bike (known tags) and students who carpool or ride the bus (unknown tags). The system only records a known tag once per day in order to offer incentives to students who walk and bike, but records unknown tags as the same student ID in order to obtain data for students who carpool or ride the bus as well. The LEDs confirm when the system logs a tag. A more detailed description of the system operation is located in Section 4 of Appendix F.

Program revision process

The initial prototype stored 20 known tags on the microcontroller in order to test the capacity when interacting with the reader. After the initial program was successful, the number of tags was gradually increased until the limit of 110 tags (due to the amount of memory available on the microcontroller) was reached. Next, the program was adapted to write confirmation of a known tag to the microSD card. To allow the end user to edit the list of known tags more easily, the program was altered to read a list from a file on the microSD card. Originally, the limit of 110 was enough to track the students who walk or bike to school, but after feedback from the Safe Routes program, the system was expanded to log unknown tags, which would be

used to track students who carpool or bus to school. A simple program to allow identification of new tags was also written to benefit the end user. The code for both programs is listed in Appendix E.

Programming logic

When the system is powered on, the microcontroller goes through the process illustrated in Figure 6. Full details of what each function does are located in the comments within the code in Appendix E.

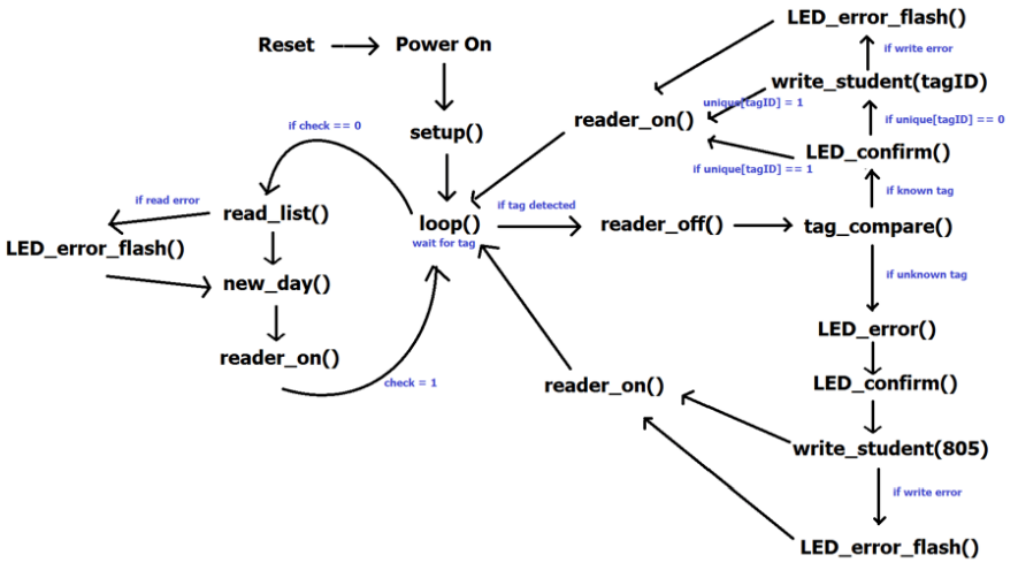


Figure 6: srts_final flow chart

Below, Figure 7 illustrates the logic of the tag identification program (new_tag), which can be uploaded to the microcontroller as necessary in order to identify new tags. Full details of what each function does are located in the comments within the code in Appendix E. As noted in Appendix E, the base code for the reader and SD card interaction was adapted from the open source examples available on the Arduino website [7].

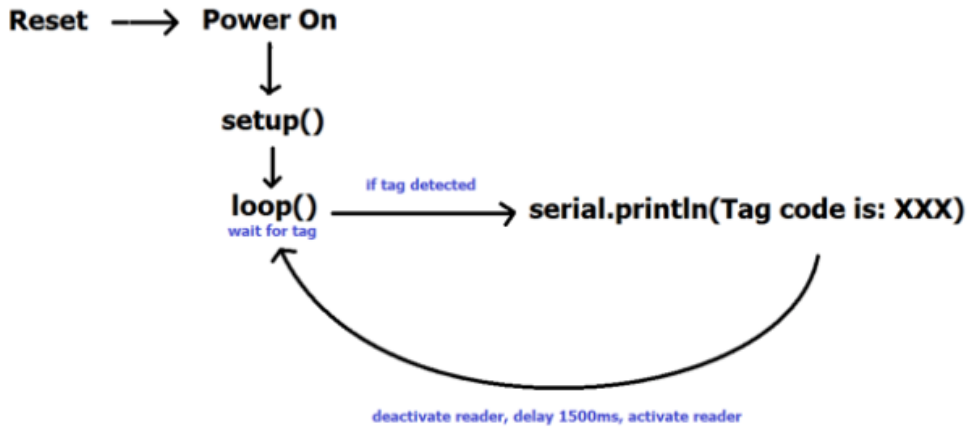


Figure 7: new_tag flow chart

V. Testing

Test Day 1

During the first test conducted at Hawthorne Elementary on May 23rd, 43 students received tags and were successfully logged during arrival to school. The first test revealed that signage was needed on the device to improve student understanding, so simple directions were added to the cover of the encasement (see Figure 10). It is also important to note that students were naturally curious about the system, and were excited to use it. Figure 8 shows students lining up to register and receive their RFID tags.

Figure 8: Students registering with the system (photo credit: Dr. Benson)

Test Day 2

During the second test day on May 30th, over 70 students were logged with the system, however because it was also National Bike Day, the Safe Routes program was handing out free Jamba Juice and t-shirts. This led to only brief interactions with the system from new students; however, many students who had previously received tags remembered to bring them back and use them again. The new signage also seemed to lower the amount of confusion with the device, though it was hard to judge because of the other activities of the morning. Tags handed out during the first test were collected for use during the next school year.

VI. Conclusion

As discussed in Section V, the system successfully logged every student that biked and walked to school during testing. The system meets all goals and requirements (as outlined in Section III) of the Safe Routes program for a low-cost method of collecting data on the number of students who walk, bike, carpool, and bus to school. The total cost of the components was roughly $250 (Appendix C, Table I), which was much lower than originally projected. The system will be turned over to the Safe Routes program before the start of the next school year. The program hopes to begin collecting data next fall. A User Manual (Appendix F) was created for the reference of the program during data collection next year. Future directions could include duplicating the system for use at other schools that wish to collect data for their Safe Routes programs, or an automated RFID system (as discussed in Section II) if the program is able to receive more funding.

VII. Bibliography

[1] Boltage, Inc. (2011). *How it works.* Available: http://www.boltage.org/how.html

[2] National Center for Safe Routes to School. *About us.* [Online]. Available: http://www.saferoutespartnership.org/

[3] National Center for Safe Routes to School (November 2011). *How Children get to School.* [Online]. Available: http://www.saferoutesinfo.org/program-tools/NHTS-school-travel-1969-2009

[4] Parallax, "RFID Card Reader, Serial (#28140)," datasheet, August 2008.

[5] Sanghera, Paul, *CompTIA RFID+.* Rockland, MA: Syngress Publishing, Inc., 2007.

[6] Source for images: http://www.parallax.com/

[7] Source for images: http://www.arduino.cc/

VIII. Appendices

Appendix A - Senior Project Analysis

Project Title: RFID Check-in System

Student's Name: Paul Banel **Student Signature:**

Advisor's Name: Bridget Benson **Advisor's Initials:** **Date:**

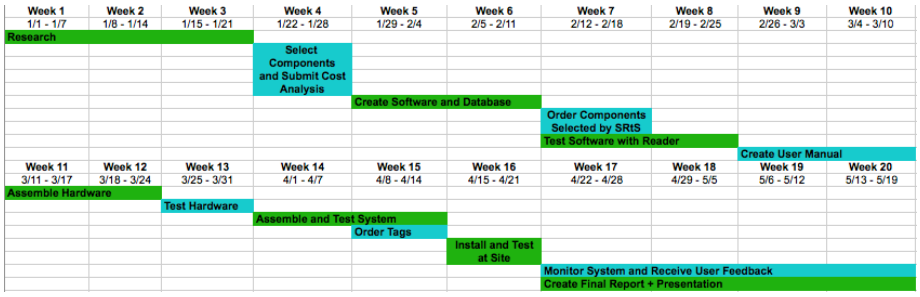
Summary of Functional Requirements
The system uses passive RFID tags that students at Hawthorne Elementary will receive next year in order to track the number of students who walk and bike to school for the Safe Routes to School program. The program hopes to use this data to apply for funding from the Safe Routes to School National Partnership to use to improve routes to school and offer incentives for students to walk and bike.

Primary Constraints
During the first half of the project life cycle communication with the Safe Routes program was difficult: Email response time was lengthy and I had no other way of contacting the program, so the exact requirements for the system (based on the needs of the program) were not known until much later than expected. Another difficulty encountered with the code was that the Arduino development environment had some issues when using certain functions in certain expected ways. These issues were eventually resolved without knowing what caused the nuances.

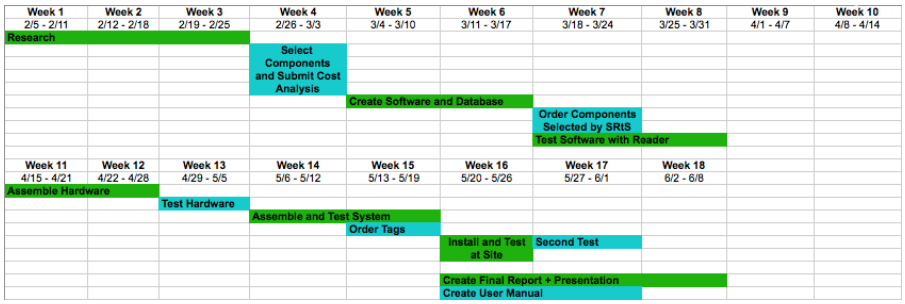
Economic
The original cost of the system was estimated to be close to $800 due the original expectation of many more students than there actually are at Hawthorne, and a permanent outdoor structure for the system to be enclosed in. This was due in part to lack of communication from the program, but I also did not want to underestimate the costs. The final cost came out to be $250. Additional costs will occur if new tags need to be ordered to add new students or replace any tags lost by students participating in the program.

Any economic benefits for the Safe Routes program will depend on how much data they collect, and how successful the program is at encouraging children to walk and bike to school. These factors will determine if and how much grant money they can receive from the national program.

The original timing was shifted to align better with the Safe Routes availability for testing at Hawthorne. The two Gantt charts below show how the timing of the project was changed.



Original Timeline



Revised Timeline

Even though the senior project side of the system is complete, the system will be implemented and used at Hawthorne Elementary next year to collect data for the Safe Routes program. I will be available to answer any questions, should the program have any, next fall.

Environmental

The use of this system will encourage students to walk and bike to school, and should have a net positive environmental impact by reducing the amount of vehicle traffic to and from Hawthorne. One possible negative impact is that if students misplace the tags (which is feasible for the age group and size of the tag) is that it creates unnecessary waste.

Manufacturability

Since the system is designed from commercially available hardware, it is easily duplicated for use by other Safe Routes programs.

Sustainability

A long-term durability issue is that the current encasement is easily tampered with, however since its use will be monitored during use, this is not a large setback. A permanent encasement for the system at Hawthorne could be developed, however this would make it difficult to extract data from the device, as well as use it to identify new tags, which is currently a simple task in its portable form.

Ethical

Since participation in the program is voluntary and requires an interaction from the user with the tag, no ethical implications exist with its use. However, if it were upgraded to an automated system, this could begin to infringe on privacy if students are unaware that they are being monitored on their way to and from school.

Health and Safety
The system is potentially beneficial to the health of students, as its use will encourage children to walk and bike to school more often, which will improve their health in the long run. Since it operates at a low voltage, uses a very low power wireless communication, and is self-contained, there are no safety risks when operating or interacting with the system.

Social and Political
The goal for use of the system is to use data collected to receive funds to improve the safety of the community, as well as improve the long-term health of students. In this sense, the stakeholders are all community members, but more specifically the students and their parents. Only students who walk and bike to school will directly benefit from any improvements made through use of the system, however because the goal is to also encourage more children to walk and bike, the majority of students could potentially benefit during the life of the system.

Development
One of the main new tools I gained from this project was learning to use the Arduino programming environment, which I can use for future projects. I also gained valuable experience working with a real client in the community, and learning how to work best around their schedule realistically mimics engineering projects I could experience in industry.

Appendix B – Project Timeline

The original timeline of the project (shown in Figure 9) was modified slightly due to communication delays with the Safe Routes committee. Overall, the timeline was shifted forward by approximately five weeks after it was determined that the initial test would be performed on 5/23. The order of events stayed mostly the same, with the exception of the creation of the user manual occurring after the initial test and feedback.

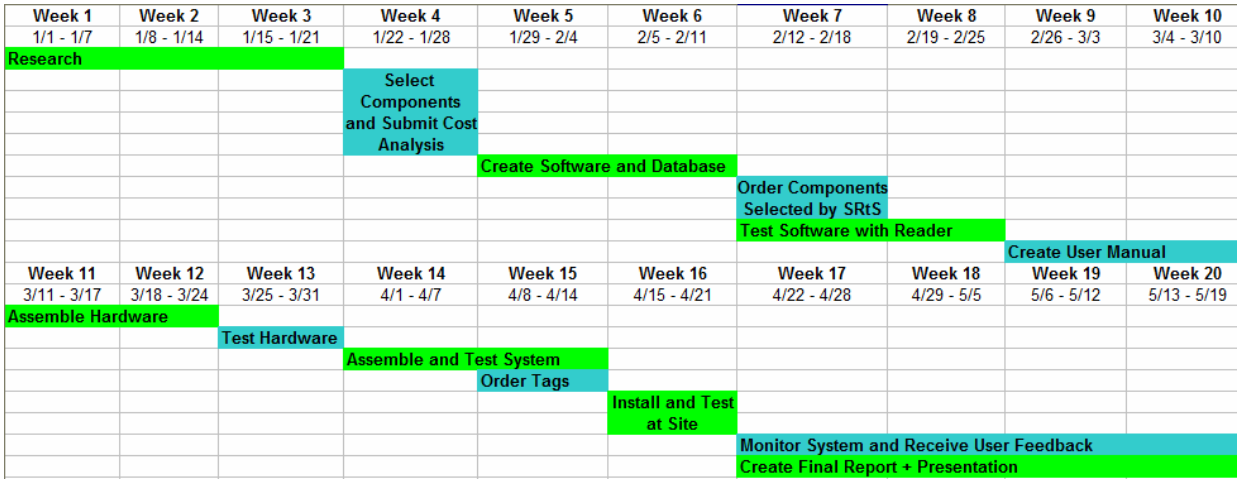


Figure 9: Original project timeline

Appendix C – Parts List and Cost

Table I shows the list of all parts used in the final system with their approximate costs. Items marked with a * were recycled from the author's previous projects to keep costs low.

Table I: PARTS LIST AND COST

Part	**Cost**
Arduino SD Shield	$35
*Arduino Uno SMD	$30
microSD Card and Adapter	$15
*Parallax RFID Reader (serial)	$40
Parallax 54x85mm Rectangle Tags (120)	$95
Plastic Container, 6x8"	$5
*USB A to USB B Cable, 6ft	$5
USB AC Adapter	$20
*Misc. Components (LEDs, Resistors, Wires)	<$1
	Total: $246

Appendix D – Hardware and Connections

Hardware components

This section contains photos and diagrams of the hardware, referenced in Section IV of this report.

Figure 10: External view of system enclosure

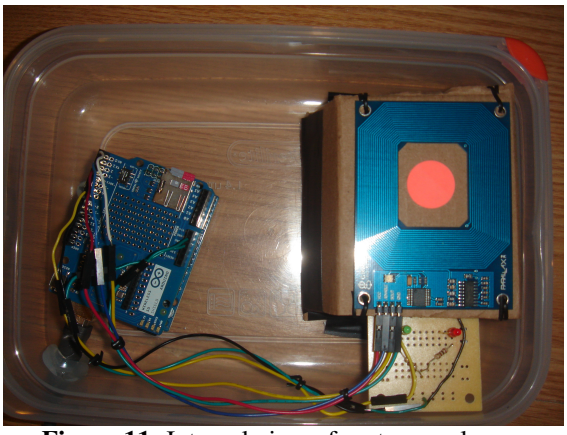


Figure 11: Internal view of system enclosure

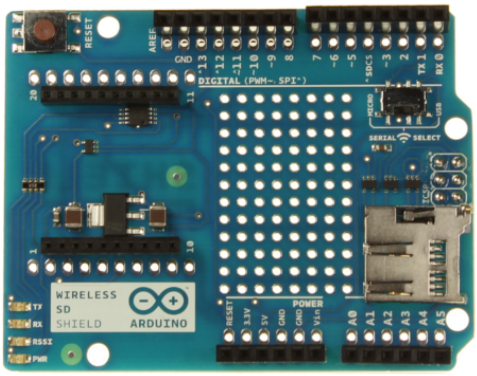


Figure 12: Arduino Uno SMD [6]

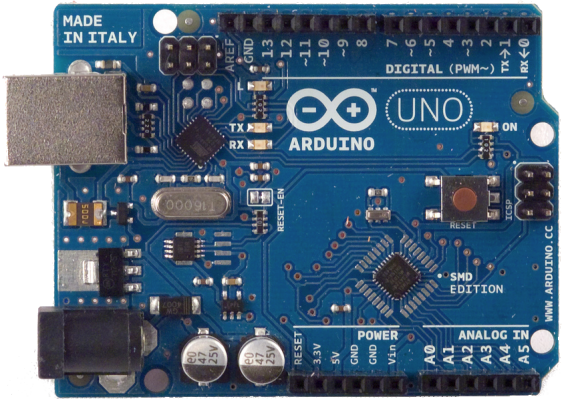


Figure 13: Arduino Wireless SD Shield [6]

Figure 14: Parallax RFID Reader (serial) and 54x85mm tags [7]

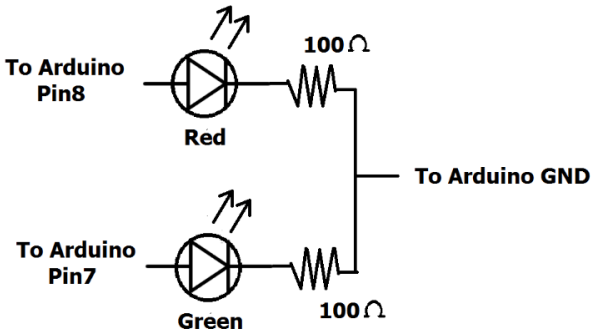


Figure 15: LED module

Wire connections

The system connects in the following manner, illustrated in Figure 16. The Arduino Wireless SD Shield connects directly onto the Uno, and uses the same pin connections.

- Arduino Pin0 (RX) to Parallax SOUT (white wire)
- Arduino Pin2 to Parallax /ENABLE (blue wire)
- Arduino Pin3 to Parallax VCC (red wire)
- Arduino Pin7 to Green LED (yellow wire)
- Arduino Pin8 to Red LED (black wire)
- Arduino GND to Parallax GND (green wire)
- Arduino GND to LED GND (green wire)

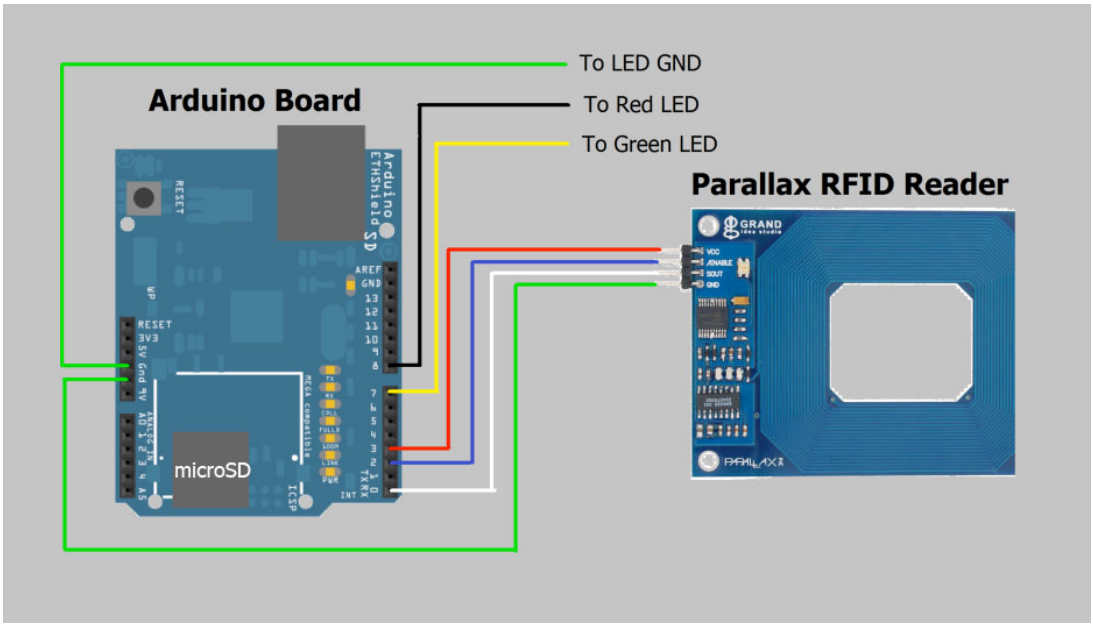


Figure 16: Wire connections [6], [7]

Appendix E – Code

Arduino code for "srts_final"

This is the main program for the system and handles all operations discussed previously in this report.

```
/*****************************************************
* File: srts_final.ino
*
* Author: Paul Banel
* Advisor:  Bridget Benson
* EE Senior Project
* Cal Poly SLO
* Spring 2012
*
* RFID Safe Routes to School Check-in System
*
* Reads and tracks unique RFID tags that users scan
* in order to track the number of students who participate
* in the Safe Routes to School program at Hawthorne
* Elementary.  110 unique tags allow for tracking both
* the total number of participants per day as well as how
* many times each unique tag is scanned over the the course
* of many days.
*
*
* Revisions:
*
* ver1:  Stored 20 tags on the board to compare with
*        tags from reader.
*
* ver2:  Read 20 tags from text file on SD card to
*        compare with rags from reader.
*
* ver3:  Read 20 tags from text file on SD card,
*        stored in single array for comparison.
*
* ver4:  Read from SD card until end of text file.
*        Goal was to store up to 100 tags in a single
*        array for comparsion, but required too much memory
*        for the size of the board.
*
* ver5:  Read set number of tags (100) from text file,
*        stored in individual strings for comparison.
*        Compares last 3 hex numbers of tags.
*
* ver6:  Separated components of loop() function into
*        smaller functions for better structure.  Tag
*        list increased to 110 from 100.  110 determined
*        to be upper limit for comparing the last 3 hex
*        digits of the tags.
*
* ver6.1:  Attempted to increase number of tags the system
*          could store by comparing only the last 2 hex
*          digits on the tag. Only increased tag list to 130
*          but invalidated some tags due to duplicates
*          (e.g. 0A4 and FA4 became the same tag), so
*          this method was abandoned.
*
* Final:  Known tag limit is atill 110, however unknon tags

*         allowed to be tracked for purpose of recording
*         the total number of users in case the program
*         exceeds 110.  Minor edits to code flow.
```

```
*
* Components:  Arduino Uno, Arduino Wireless SD Shield,
* Parallax RFID Reader (serial), red LED, green LED,
* 2 100 ohm Rs
*
* ----------HARDWARE CONNECTIONS--------------------
* Uno RX to Parallax SOUT  ***DISCONNECT BEFORE UPLOADING
* Uno PIN2 to Parallax /ENABLE
* Uno PIN3 to PARALLAX VCC
* Uno PIN7 to Green LED to 100 ohm to GND
* Uno PIN8 to Red LED to 100 ohm to GND
* Uno GND to Parallax GND
*
*
* RFID code adapted from arduino.cc/playground/Learning/PRFID
* SD code adapted from arduino.cc/en/Tutorial/ReadWrite
*
*******************************************************/

#include <SD.h>

//file that contains known tags
File tag_list;
//file that logs students
File rfid_log;


//variables used with RFID reader
int val = 0;
//temporary storage for tag number
char code[10];
//determines which byte is stored
int bytesread = 0;

//List of empty tags to be read from file
char stu0[4] = "000";
char stu1[4] = "000";
char stu2[4] = "000";
char stu3[4] = "000";
char stu4[4] = "000";
char stu5[4] = "000";
char stu6[4] = "000";
char stu7[4] = "000";
char stu8[4] = "000";
char stu9[4] = "000";
```

*****NOTE: char definitions for tags 10 through 109 follow the same format as 0 through 9, and were omitted to conserve space*****

```
//used to read unneeded bytes from tag_list
char empty = '0';
//flag to determine if tag has been logged yet
int unique[110];
//setup flag
int check = 0;

/*--------------------------------------------------------
* Function: setup
*
* Description: Sets the pins used in the program to the
* necessary modes and set up for SD library.
*
*
*--------------------------------------------------------*/
void setup()
{
```

```
  //baud rate for SD card
  Serial.begin(9600);
  //designates pin 2 as output for /ENABLE on reader
  pinMode(2, OUTPUT);
  //pin 3 used to turn reader on/off
  pinMode(3, OUTPUT);
  //pin 7 for confirm LED
  pinMode(7, OUTPUT);
  //pin 8 for error LED
  pinMode(8, OUTPUT);
  //allows use of SD library
  pinMode(10, OUTPUT);
  //initialize reader disabled
  digitalWrite(2, LOW);
  //initialize reader off
  digitalWrite(3, LOW);
  //initialize with LEDs off
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);

  //necessary for reading/writing from/to sd card
  if (!SD.begin(4))
  {
    LED_error_flash();
    return;
  }
}

/*-------------------------------------------------------
* Function: loop
*
* Description: Arduino equivalent of main funtion. After
* additional set up, waits for tag to be detected by
* and reacts depending on tag.  If the tag is known, log
* the associated student number.  If the tag is unknown,
* log as unregistered student.
*
* Notes for Additonal setup:
*
* Reads and stores known tag list from SD card.  Sets all
* unique flags to 0.  Prints a break line in log file
* to distinguish between days.  Activiates reader and exits
* to main loop. If these steps are performed in the "setup()"
* function, it causes the device to not function properly.
* The cause is still unknown after troubleshooting.
*
*-------------------------------------------------------*/
void loop()
{
  //Additonal setup.
  if(check == 0)
  {
    read_list();

    //initialize unique flags to 0
    for (int k = 0; k < 110; k++)
      unique[k] = 0;

    new_day();
    check = 1;
    delay(1000);
    reader_on();
  }

  //loops until tag is detected
  if(Serial.available() > 0)
  {
```

```
    //read code from tag
    if((val = Serial.read()) == 10)
    {
      bytesread = 0;
      while(bytesread < 10)
      {
        if(Serial.available() > 0)
        {
          val = Serial.read();
          //if tag read is incomplete or finishes too quickly, end reading
          if((val == 10) || (val == 13))
          {
            break;
          }
          //stores current byte from tag
          code[bytesread] = val;
          //increment to next byte
          bytesread++;
        }
      }

      if(bytesread == 10)
      {
        reader_off();
        tag_compare();
      }

      bytesread = 0;
      delay(1000);
      reader_on();
    }
  }
}

/*-----------------------------------------------------
* Function: LED_confirm
*
* Description: Turns on green LED for one second.  Used
* as a confirmation of known tag.
*
*-----------------------------------------------------*/
void LED_confirm()
{
  digitalWrite(7, HIGH);
  delay(1000);
  digitalWrite(7, LOW);
}

/*-----------------------------------------------------
* Function: LED_error
*
* Description: Turns on red LED for one second.  Used
* as a confirmation of unknown tag.
*
*-----------------------------------------------------*/
void LED_error()
{
  digitalWrite(8, HIGH);
  delay(1000);
  digitalWrite(8, LOW);
}

/*-----------------------------------------------------
* Function: LED_error_flash
*
* Description: Flashes red LED to indicate an error when
* reading from or writing to SD card.
```

```
*
*-------------------------------------------------------*/
void LED_error_flash()
{
   for (int i = 0; i < 3; i++)
   {
     digitalWrite(8, HIGH);
     delay(500);
     digitalWrite(8, LOW);
     delay(500);
   }
}

/*-------------------------------------------------------
* Function: new_day
*
* Description: Prints "-----New Day-----" to log file
* on SD card to indicate a new log session.
*
*-------------------------------------------------------*/
void new_day()
{
   //set baud rate for SD card
   Serial.begin(9600);
   rfid_log = SD.open("rfid.txt", FILE_WRITE);
   if (rfid_log)
   {
     rfid_log.println("-----New Day-----");
     rfid_log.close();
   }
   else
     LED_error_flash();
//set baud rate for reader
   Serial.begin(2400);
}

/*-------------------------------------------------------
* Function: reader_off
*
* Description: Deactivates reader (active low) and turns
* reader off.
*
*-------------------------------------------------------*/
void reader_off()
{
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
}

/*-------------------------------------------------------
* Function: reader_on
*
* Description: Turns reader on and activates reader.
*
*-------------------------------------------------------*/
void reader_on()
{
  digitalWrite(3, HIGH);
  digitalWrite(2, LOW);
}

/*-------------------------------------------------------
* Function: read_list
*
* Description: Reads text file containing known tags from
* SD card. Each tag is separated by two reads into the
* char empty to accomodate for null and line breaks after
```

```
* each tag ID in the text file.  Comments every 10 tags
* divide the function into sections for convenience of
* author.
*
*-------------------------------------------------------*/
void read_list()
{
  int i = 0;
  Serial.begin(9600);
  tag_list = SD.open("taglist.txt", FILE_READ);

  if (tag_list)
  {
    //tags 0 through 9
    stu0[0] = tag_list.read();
    stu0[1] = tag_list.read();
    stu0[2] = tag_list.read();
    empty = tag_list.read();
    empty = tag_list.read();
    stu1[0] = tag_list.read();
    stu1[1] = tag_list.read();
    stu1[2] = tag_list.read();
    empty = tag_list.read();
    empty = tag_list.read();
    stu2[0] = tag_list.read();
    stu2[1] = tag_list.read();
    stu2[2] = tag_list.read();
    empty = tag_list.read();
    empty = tag_list.read();
    stu3[0] = tag_list.read();
    stu3[1] = tag_list.read();
    stu3[2] = tag_list.read();
    empty = tag_list.read();
    empty = tag_list.read();
    stu4[0] = tag_list.read();
    stu4[1] = tag_list.read();
    stu4[2] = tag_list.read();
    empty = tag_list.read();
    empty = tag_list.read();
    stu5[0] = tag_list.read();
    stu5[1] = tag_list.read();
    stu5[2] = tag_list.read();
    empty = tag_list.read();
    empty = tag_list.read();
    stu6[0] = tag_list.read();
    stu6[1] = tag_list.read();
    stu6[2] = tag_list.read();
    empty = tag_list.read();
    empty = tag_list.read();
    stu7[0] = tag_list.read();
    stu7[1] = tag_list.read();
    stu7[2] = tag_list.read();
    empty = tag_list.read();
    empty = tag_list.read();
    stu8[0] = tag_list.read();
    stu8[1] = tag_list.read();
    stu8[2] = tag_list.read();
    empty = tag_list.read();
    empty = tag_list.read();
    stu9[0] = tag_list.read();
    stu9[1] = tag_list.read();
    stu9[2] = tag_list.read();
    empty = tag_list.read();
    empty = tag_list.read();
```

*****NOTE: Operations for tags 10 through 109 follow the same format as 0 through 9, and were omitted to conserve space*****

```
   tag_list.close();
  }
  else
  {
   LED_error_flash();
  }
  Serial.begin(2400);
}

/*-------------------------------------------------------
* Function: tag_compare
*
* Description: Compares the most recent tag read from
* the reader with the list of known tags.  If known,
* use function "write_student".  If unknown, use function
* "uknown_student".  Function is commented every 10 tags
* for convenience of author.
*
*-------------------------------------------------------*/
void tag_compare()
{
      //tags 0 through 9
      if(code[7] == stu0[0] && code[8] == stu0[1] && code[9] == stu0[2])
      {
       if(unique[0] == 0)
         write_student(0);
       else
         LED_confirm();
      }
      else if(code[7] == stu1[0] && code[8] == stu1[1] && code[9] == stu1[2])
      {
       if(unique[1] == 0)
          write_student(1);
       else
          LED_confirm();
      }
      else if(code[7] == stu2[0] && code[8] == stu2[1] && code[9] == stu2[2])
      {
       if(unique[2] == 0)
         write_student(2);
       else
         LED_confirm();
      }
      else if(code[7] == stu3[0] && code[8] == stu3[1] && code[9] == stu3[2])
      {
       if(unique[3] == 0)
         write_student(3);
       else
         LED_confirm();
      }
      else if(code[7] == stu4[0] && code[8] == stu4[1] && code[9] == stu4[2])
      {
       if(unique[4] == 0)
          write_student(4);
       else
          LED_confirm();
      }
      else if(code[7] == stu5[0] && code[8] == stu5[1] && code[9] == stu5[2])
      {
       if(unique[5] == 0)
          write_student(5);
       else
          LED_confirm();
```

```
      }
      else if(code[7] == stu6[0] && code[8] == stu6[1] && code[9] == stu6[2])
      {
       if(unique[6] == 0)
          write_student(6);
       else
          LED_confirm();
      }
      else if(code[7] == stu7[0] && code[8] == stu7[1] && code[9] == stu7[2])
      {
       if(unique[7] == 0)
          write_student(7);
       else
          LED_confirm();
      }
      else if(code[7] == stu8[0] && code[8] == stu8[1] && code[9] == stu8[2])
      {
       if(unique[8] == 0)
          write_student(8);
       else
          LED_confirm();
      }
      else if(code[7] == stu9[0] && code[8] == stu9[1] && code[9] == stu9[2])
      {
       if(unique[9] == 0)
          write_student(9);
       else
          LED_confirm();
      }
```

*****NOTE: Operations for tags 10 through 109 follow the same format as 0 through 9, and were omitted to conserve space*****

```
      //if unknown tag, send number outside student number range
      else
      {
       write_student(805);
      }
}

/*---------------------------------------------------------
* Function: write_student
*
* Description: Writes the student number associated with
* a known tag and sets the tag's unique flag to 1. Turns
* on green LED to indicate known tag. If unknown tag,
* write "Student805" and turn on red LED, then green LED
* to indicate unknown tag.
*
* param stu_num: int: Number that corresponds with known
* tag.  Tells function to write this student number to
* the SD card.  If sent "805", tag is unknown, but logged
* for keeping track of overall number of students.
*---------------------------------------------------------*/
void write_student(int stu_num)
{
 //match baud rate of SD shield
 Serial.begin(9600);
 rfid_log = SD.open("rfid.txt", FILE_WRITE);
 if (rfid_log)
 {
   //print which student's tag was scanned to file
   rfid_log.print("Student");
   rfid_log.println(stu_num);
   rfid_log.close();
   //flag tag as read
```

```
    if (stu_num == 805)
    {
      LED_error();
      LED_confirm();
    }
    else
    {
      unique[stu_num] = 1;
      LED_confirm();
    }
  }
  else
  {
  //flash error LED if not written to file
    LED_error_flash();
  }
//change baud rate back to match reader
Serial.begin(2400);
}
```

Arduino code for "new_tag"

Since Parallax does not ship new tags with their ID number, this program allows the user to determine the ID numbers of a new RFID tag in order to update the known tags.

```
/*****************************************************
* File: new_tag.ino
*
* Author: Paul Banel
* Advisor:  Bridget Benson
* EE Senior Project
* Cal Poly SLO
* Spring 2012
*
* RFID Safe Routes to School Check-in System
* New tag identification
*
* Reads and prints the last 3 hex numbers of the
* RFID tag to the Serial Monitor. Used to identify
* new tags, as Parallax does not print or identify
* new tags that are used with the reader.
*
*
* Components:  Arduino Uno, Arduino Wireless SD Shield,
* Parallax RFID Reader (serial), red LED, green LED,
* 2 100 ohm Rs
*
* ----------------HARDWARE CONNECTIONS--------------------
* Uno RX to Parallax SOUT  ***DISCONNECT BEFORE UPLOADING
* Uno PIN2 to Parallax /ENABLE
* Uno PIN3 to PARALLAX VCC
* Uno PIN7 to Green LED to 100 ohm to GND
* Uno PIN8 to Red LED to 100 ohm to GND
* Uno GND to Parallax GND
*
*
* RFID code adapted from arduino.cc/playground/Learning/PRFID
*
*****************************************************/

int  val = 0;
char code[10];
int bytesread = 0;

/*-----------------------------------------------------
* Function: setup
```

```
*
* Description: Sets baud rate of board to match the reader
* and all pins to necessary settings.
*
*-------------------------------------------------------------*/
void setup()
{
  //baud rate of reader
  Serial.begin(2400);
 //pin connected to /ENABLE on reader
  pinMode(2,OUTPUT);
  //pin connected to Vcc on reader
  pinMode(3,OUTPUT);
  //turn reader on
  digitalWrite(3, HIGH);
  //enable reader (active low)
  digitalWrite(2, LOW);
}

/*-------------------------------------------------------
* Function: loop
*
* Description: Arduino equivalent of main function. Waits
* for tag to be detected by reader, then scans and stores
* each byte of tag.  Prints the last 3 bytes of the tag
* to the Serial Monitor.
*
*
*-------------------------------------------------------------*/
void loop()
{
  //wait for tag to be detected
  if(Serial.available() > 0) {
    if((val = Serial.read()) == 10) {
      bytesread = 0;
      while(bytesread<10) {
        if( Serial.available() > 0) {
          val = Serial.read();
          if((val == 10)||(val == 13)) {
            break;
          }
          code[bytesread] = val;
          bytesread++;
        }
      }
      //print "Tag code is: [last 3 hex of tag]
      if(bytesread == 10)
      {
        Serial.print("TAG code is: ");
        Serial.print(code[7]);
        Serial.print(code[8]);
        Serial.println(code[9]);
      }

      //deactivate reader and pause before next read
      bytesread = 0;
      digitalWrite(2, HIGH);
      delay(1500);
      digitalWrite(2, LOW);
    }
  }
}
```

Appendix F – User Manual

RFID Check-in System

for Safe Routes to School at Hawthorne Elementary

User Manual

Paul Banel
California Polytechnic State University
San Luis Obispo
Spring 2012

1 - Setting up the software

If this is the first time using the system with a new computer, this section will walk the user through the steps required to do so.

1.1 - Installing the software and drivers

1. Go to http://arduino.cc/en/Main/Software
2. Select the appropriate download based on your operating system.
3. After downloading the file, unzip in desired directory.
4. Connect the USB cable from the device to the computer.

For Windows operating systems:

5a. Go to http://arduino.cc/en/Guide/Windows#toc4 and follow the step-by-step instructions to install the driver.

For Mac OS X operating systems:

5b. Go to http://arduino.cc/en/Guide/MacOSX#toc4 and follow the instructions in Section 4 "Connect the Board".

1.2 - Creating the project files

6. From where you unzipped the files, open the arduino.exe file (arduino).
7. Start a new file (File --> New).
8. Open "srts_final.txt" from the "Data" folder on the microSD card (see section 5.2) and copy the entire text into the new Arduino file.
9. Save this file (File --> Save As…) as "srts_final" in the default location.
10. Start a new file (File --> New).
11. Open "new_tag.txt" from the "Data" folder on the microSD card and copy the entire text into the Arduino new Arduino file.
12. Save this file (File --> Save As…) as "new_tag" in the default location.

For troubleshooting or additional questions regarding setting up the software or board, go to http://arduino.cc/en/Guide/HomePage and select your operating system from the list.

2 - Adding or changing tag ID numbers

This section discusses how to add or change tag ID numbers to the list of known tags.

2.1 - Preparing the device

1. Open the arduino.exe file to start the software (arduino).
2. Open the file "new_tag.ino" (File --> Open).
3. Connect the USB cord from the device to the computer.
4. Disconnect the white wire from the Arduino RX port (see section 5.1).
5. Upload the file to the device (File --> Upload). A log at the bottom of the window will show the upload progress.
6. Once the file has finished uploading, reconnect the white wire to the Arduino RX port.
7. Open the Serial Monitor (Tools --> Serial Monitor).
8. Set the baud rate to 2400 baud (lower right hand corner of the Serial Monitor window).

2.2 - Reading new tags

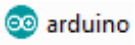
9. Wave new tags over the reader. The last 3 numbers of the tag should display on the Serial Monitor.

2.3 - Adding tags to the list

10. Remove the microSD card from the Arduino board (see section 5.2).
11. Place the microSD card into the SD adapter.
12. Connect the adapter to your computer.
13. Open the file "tag_list.xls" on the microSD card.
14. Add or edit any of the new ID numbers into the "3 LSB" column of the spreadsheet.
15. Save the file (File --> Save).
16. Open the file "taglist.txt" on the main level of the microSD card.
17. Delete all of the text from the file.
18. Highlight and copy (Edit --> Copy) the entire column of ID numbers from the "3 LSB" column of the Excel spreadsheet.
    - 18a. Make sure that the line "3 LSB" does not appear at the top.
    - 18b. See the file "taglist_example.txt" for proper formatting.
19. Paste the ID numbers into the file "taglist.txt" (Edit --> Paste).
20. Save the file (File --> Save).
21. Close the files "taglist.txt" and "tag_list.xls".
22. Eject the SD card adapter from your computer.
23. Remove the microSD card from the adapter.
24. Place the microSD card in the Arduino board.

3 - Setting up the device to log students

This section discusses how to prepare the device to log students.

1. Open the arduino.exe file to start the software ().
2. Open the file "srts_final.ino" (File --> Open).
3. Connect the USB cord from the device to the computer.
4. Disconnect the white wire from the Arduino RX port (see section 5.1).
5. Make sure the microSD card is connected to the Arduino board (see the "SD Card" section in the "Hardware" section).
6. Upload the file to the device (File --> Upload). A log at the bottom of the window will show the upload progress.
7. Once the file has finished uploading, reconnect the white wire to the Arduino RX port.
8. The device is now ready to log students.

4 – Device functioning

This section discusses the basic operations of the system.

4.1 – Logging students

Every time the device is plugged in or reset, the line "-----New Day-----" is printed to the log file "RFID.txt" on the main level of the microSD card (see section 5.2) to signify a new log has started. Until the device is unplugged or reset, the device will log each unique, registered tag once (even if it is scanned multiple times) and any unregistered tag as many times as it is scanned. The number of registered tags is limited to 110 students, of the format "Student0" through "Student109", and the tag ID number that corresponds to each student number is determined the file "taglist.txt" on the main level of the microSD card. Unregistered tags will appear on the log as "Student805". See section 2.3 for information on registering new tags or changing tag numbers. Due to the limit of 110 tags, it is recommended that tags only be registered to students that walk and bike, and unregistered tags be given to students that carpool or take the bus. Data on the log file is stored in the following format:

-----New Day-----
Student12
Student52
Student33
Student805
Student102
-----New Day-----
Student4
Student31
Student82
Student805

4.2 – Extracting log data

The log data from each session is stored on the "RFID.txt" file on the main level of the microSD card (see section 5.2). As discussed in section 4.1, the log will keep track of each unique tag that is scanned. The total number of students per day is the number of students between each "-----New Day-----" line, which can be totaled and stored in a format of the user's choosing. Each unique ID can also be recorded in the "Total Days" column of the "tag_list.xls" file in the "Data" folder on the microSD card in order to track how many times each student checks in with the system (or in another format of the user's choosing). After the data has been recorded, the entire "RFID.txt" file can be cleared in order to prevent re-using previous data, making sure that the empty "RFID.txt" file remains on the main level of the microSD card.

4.3 – Scanning tags with the device

When a student wishes to check in with the system, they can wave their tag briefly in the air over the area marked on the device. If the green LED lights up, the tag has been logged as a unique student ID (Student0 through Student109). If the red LED lights up, followed by the green LED, the tag has been logged as an unregistered user (Student805). This is also useful for testing tags after adding or changing IDs, as discussed in section 2.

5 - Hardware

This section discusses the different hardware components of the system.

5.1 - Arduino board and Parallax RFID reader

The Arduino board handles all of the data writing and storage used by the system. The Parallax RFID reader recognizes the tags and tells the Arduino board which tag it reads. The following connections are needed for the device to function properly, and are illustrated in **Figure 1** below:

- Arduino Pin0 (RX) to Parallax SOUT (white wire)
- Arduino Pin2 to Parallax /ENABLE (blue wire)
- Arduino Pin3 to Parallax VCC (red wire)
- Arduino Pin7 to Green LED (yellow wire)
- Arduino Pin8 to Red LED (black wire)
- Arduino GND to Parallax GND (green wire)
- Arduino GND to LED GND (green wire)

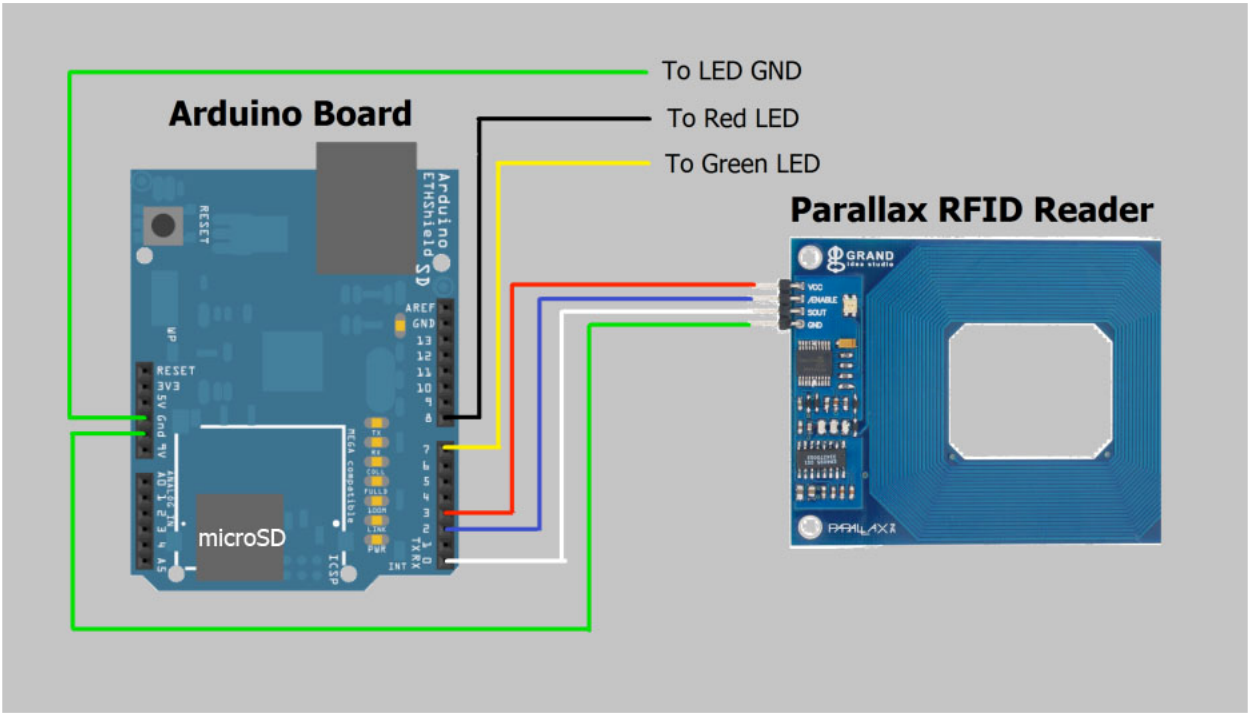


Figure 1: Hardware Connection Diagram

5.2 - SD Card

The microSD card is used to store the known tags and log students when they use the device. To place or remove the card from the Arduino board, push in once to lock in place, and push once more to eject. The microSD card can be placed in the SD adapter to connect with a

computer to add or read files. The card contains the main level, where the files "taglist.txt" and "RFID.txt" files are stored (see sections 1 and 4), and the "Data" folder where all other files are stored (see sections 1 and 2). A microSD card and SD adapter are shown in **Figure 2**.

Figure 2: microSD card and adapter

5.3 - Parallax RFID tags

The system uses passive RFID tags from Parallax. While there are other versions of the tags available (such as key chains), 54mm x 85mm tags (shown in **Figure 3**) have been tested and found to be the most reliable. Currently, Parallax does not identify the ID numbers of new tags before shipping them. For information on determining the ID of new tags and changing or adding tags to the device, see section 2. New RFID tags can be ordered from:

http://www.parallax.com/Store/Accessories/Hardware/tabid/162/ProductID/115/List/1/Default.aspx?SortField=ProductName,ProductName

Figure 3: Parallax 54mm x 85mm RFID Tag

6 - Troubleshooting

Many errors can be corrected by following the steps in section 3 to re-upload the program to the device, however some errors that this may not fix are discussed in more detail in this section.

6.1 - Blinking red LED

The red LED blinks multiple times at some point during operation. This means there has been an error either writing to or reading from the microSD card.

Possible causes

1. The connection to the microSD card may be poor. Try removing then replacing the microSD card (see section 5.2), then press the Reset button on the Arduino board to reset the system.

2. The files on the microSD card may be missing or incorrectly formatted. Check the files on the microSD card to make sure that both the file "RFID.txt" and "taglist.txt" are in the main level of the card (see section 5.2). Also check to makes that "taglist.txt" is the same format as "taglist_example.txt" (in the "Data" folder on the microSD card).

3. If the previous two steps do not resolve the issue, the program may not have uploaded correctly to the Arduino board, or is malfunctioning. Follow the steps outlined in section 3 in order to re-upload the program.

6.2 - Tags fail to read or read incorrectly

A tag either does not read or reads incorrectly, such as a registered tag reading as an unregistered rag (see section 4.3).

Possible causes

1. The connection to Pin0 on the Arduino board may be poor or disconnected. Make sure that Pin0 on the Arduino board is connected to the /SOUT pin of the Parallax RFID reader (see section 5.1).

2. The tag may not be in the "taglist.txt" file or the formatting of the file may be incorrect. To fix this, follow steps 10 through 24 in section 2.3. If it is a new tag, follow all the steps outlined in that section.

3. If the previous two steps do no resolve the issue, the tag may be damaged, and may need to be replaced.

For any other issues or questions, please contact the designer and author, Paul Banel (**pauljbanel@gmail.com**), or his advisor, Dr. Bridget Benson (**bbenson@calpoly.edu**).