Arduino Based RFID Attendance System

A Technological Solution to a Pen and Paper Problem

By

Rick Bielski

rbielski@hawk.iit.edu

James Jerger

jjerger@hawk.iit.edu

Kevin Zheng

Kzheng8@hawk.iit.edu

ITMT492

Spring 2015

Every faculty member has at some point struggled with the tedious task of keeping track of their students' attendance. Some departments require this and most require all lower-level classes to keep attendance records. In a university setting, how should this be done? Obviously an instructor cannot shout all the students' names like a high school style roll call. In an auditorium class with over a hundred students, this would take forever not to mention the confusions that come with so many names, many of which will be unfamiliar. The 'solution' that most instructors turn to is pen and paper. Often the professor will pass around a sheet of paper for the students to write their name, ID number, and email address. An alternative to this is to have the students come sign in at the beginning of a class or at break. This method provides many opportunities for error. Maybe not all students will get to sign in, maybe a students' handwriting will be difficult for the instructor to read, or maybe the paper will become lost or destroyed. Not to mention all the opportunities these methods provide for the forging of attendance, a student could have their friend sign them in when they do not intend on attending class. And what happens with this infinite collection of sign in sheets the professor will accumulate by the end of the semester? Either the professor or the TA will be charged with manually entering the information into some sort of ad-hoc database, or they will simply stockpile these sheets, providing no clear and easy way of viewing the attendance records while providing another opportunity for the sheets to become mangled.

We have found the solution to these problems. Student ID cards have RFID transponders embedded in them. So why not just have the students tap their ID's as they walk through the classroom door? Most people would assume this would be a costly and difficult to implement system. However, using commodity parts and existing technologies we have created a package costing less than $200 wholesale and can be set up in minutes, while being tailorable to any

institution. We have created a box containing a microprocessor and RFID reader that will scan smart cards and pass the data to the cloud where it can be stored on databases. Students don't have to worry about signing in anymore. Teachers don't have to waste time entering data. Records can be searched in students. Students' friends cannot sign them in anymore. More time is used for instruction, and attendance becomes meaningful.

The Arduino uno is a small board that allows you to do many different things ranging from RFID reader to a robot. The Arduino is what you call a micro controller. "The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started." (Arduino). We utilized two digital data pins on the Arduino and the 5v power and ground pins. Our project all starts off with an RFID Reader, then the data gets sent to the Arduino, the Arduino interprets the data into something human readable. The Arduino stores the card's code in a variable and then it gets posted to our database through PHP.

First we will discuss how the Arduino works. The Arduino inventors designed the Arduino to be easy to use and write code for. Arduinos use a stripped down easy to learn and use version of C. You have two functions that are used, setup(), and loop(). The setup function lets you assign tasks to the pins and doing so tells the Arduino what to do with the data if it gets data from its pins. We are using two of the digital pins on the Arduino.

Many card readers will work, however since IIT's HawkCards are from HID Corporation, we used an HID card reader. We choose the HID Thinline II, this card reader can be

seen used across many different buildings everywhere and can we used outdoors which means it's a very rugged card reader. The card reader has many different cables coming out of it, but we're only concerned with the red, black, green, and white wires. The red and black are power and ground respectively and the green and white wires are data. The HID Card Reader requires power in the range of "5-16 VDC (Linear power supplies are recommended." ("ThinLine II Reader"). The type of data transmitted from card readers is known as Wiegand. Wiegand was actually developed by HID Corporation. Wiegand works by sending two streams of data through the two green and white data wires. The data going through these wires is binary so a series of 1s and 0s, or base 2, which are a series of ons and offs. Open Source code was found on the internet written for the Arduino to interpret this and translates it back into base 10, which is our normal numbering system using numbers from 1 through 10. The translated code is then stored in a variable in the Arduino. ("Understanding Card Data Formats").

On the Arduino we have an Ethernet shield, to use the Ethernet shield we need to import the library for it. After we imported it, we found way online to print out the IP address we get, we wanted to setup our Arduino using DHCP because different networks will treat different devices differently, on IIT's network, the network chooses what IP it wants to give you, not you, we needed to setup DHCP. We also printed the DHCP address to Serial so we could find the IP address if needed. Through the Ethernet shield library, we are able to post data to a php file on a server. We have set it up so that the file on the server takes the card code variable and then inserts that into a database. All we have to do on the Arduino side is actually post the information. Initially I was unsure how to do this and I wasn't doing it correctly, however we eventually figured it out and was able to post the data to the webserver that will then post to the database.

In order to process the php file, a webserver is needed. For this project an AWS (amazon web services) instance is used as it requires no additional physical hardware. Since the power requirements are not very extreme, a basic instance (micro; free tier) is able to provide the necessary means of processing power at almost no cost. It also allows anyone in the group to work on the server from anywhere and backing up work easy as snapshots could be taken before major changes are made in order to revert them in case things do not work.

We chose Ubuntu for our Operating System in order to configure and run an Apache Server. This required setting up the system as per a typical web server with steps such as opening up the correct ports, configuring the firewall rules, and modifying the httpd.conf settings to allow the service to run. In addition to an Apache server, we installed MySQL to support the database needed for storing student data and retrieving the data when requested. Hosting both services on the same server allows us to use “localhost” for the SQL commands, avoiding the need to connect the two via the Internet.

In order to connect the Apache and MySQL servers, php scripts are used in conjunction with simple HTML pages to create the interface for the user as well as run all the necessary SQL commands (think CRUD logic) behind the scenes to manipulate the database. The code to process CREATE, INSERT, and DELETE commands are built into the various scripts on the site, allowing them to be used without any knowledge of or needing to use SQL on the user end. The use of such is done through a simple (if not primitive) web GUI designed and formatted using php and html.

Two tables are used in this example for storing and retrieving the data. The first stores the RFID code sent by the Arduino and a timestamp of when the code was posted. This table serves as the storage and retrieval of the timed scans, as it registers a new entry with every scanned ID. For simplicity, this table only contains three columns, as shown below:

Column	Name	Type
1	ID	INT (PRIMARY KEY)
2	cardCode	VARCHAR(5)
3	scantime	TIMESTAMP

Table 1 – Setup of Card Code table

The second table contains the data that will be used to identify individual students apart when they scan in, and is set up as depicted in the next table:

Column	Name	Type
1	ID	INT (PRIMARY KEY)
2	cardCode	INT(5)
3	firstname	VARCHAR(15)
4	lastname	VARCHAR(15)
5	studentid	VARCHAR(9)

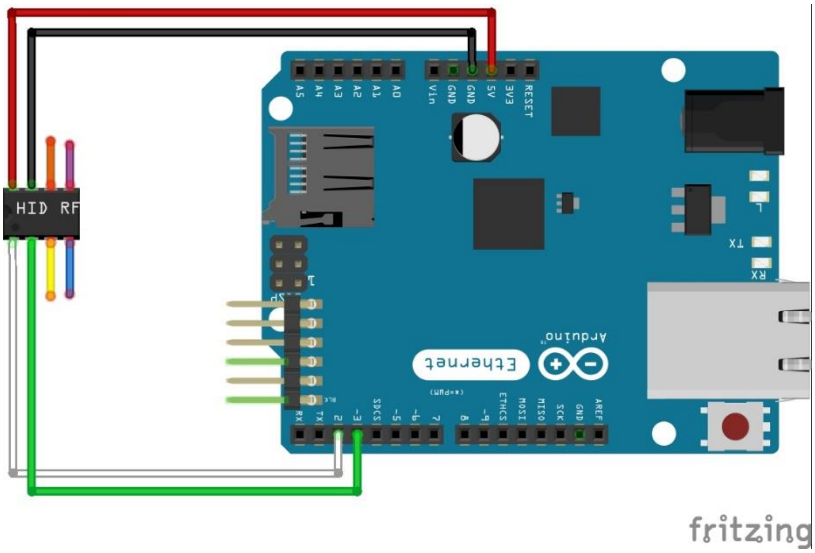
Table 2 – Setup of Students table

The server processes the code posted by the Arduino and uses the unique RFID code sent to it and uses it to check a database. How it does this is by checking the cardCodes. When a student is added to the Students table, the associated RFID code is used in this field. When a student scans in, the webpage runs a SQL query, pulling the information from both tables where the cardCode values match. This way, only the appropriate data is shown when a Student scans in. A for loop in the php code constantly checks the two tables for matching entries between the two tables and lists back the appropriate information (name, student ID, time scanned in, etc) in order from the newest to the oldest (most recent scans on top, older ones on the bottom).

The website itself that displays the information is split into multiple parts. The home page simply lists the students scanned in from most recent to oldest entry. There are a few buttons on the page that allow the user to add and delete student entries from the table. There are also two search functions: one to search for all scans between two dates and times (good for checking class attendance) and one to search all scans from a particular student. The GUI is simple and easy to use. All the code used in the site follow typical formatting convention and is easily readable and modifiable by any who have a basic understanding of both php and SQL.

While simple on the outside, the underlying code hidden to the user proved to be a real challenge. The biggest problems encountered with the project aside from the Arduino POST method were encountered at this spot in the project. There were various typos, erroneous logic errors, design change ideas, and other nuances that made completing this phase much more difficult than it had first seemed. The solution was to approach it from a software development standpoint, and go with iterative progression. This allowed us to work out problems one by one and ensure that the project was able to move along smoothly as intended. It took more time than we had originally planned but worked out in the end.

The result of all of this hard work and innovation is the end-all solution to attendance taking. The elegant one-box solution that uses off the shelf parts and open source code is infinite in its uses and abilities to track people. Today students in class, but tomorrow it can be workers clocking in or out. With a powerful database backend and a user friendly web client, paper being used for sign-ins and timecards is a thing of the past. Completely scalable and with set up that only requires plugging in two cords, anyone could benefit from our system, and be forever confident that they have the tools to track their students, or employees, or customers, or vendors, or anyone and anywhere that access tracking is critical.


HID RF
Arduino
Ethernet
fritzing

Works Cited

Arduino. "ArduinoBoardUno." Arduino - ArduinoBoardUno. Arduino, n.d. Web. 04 May 2015.

<http://www.arduino.cc/en/Main/ArduinoBoardUno>.

HID Corporation. "Understanding Card Data Formats." Understanding Card Data Formats (n.d.): n. pag. Web. 4 May 2015.

<https://www.hidglobal.com/sites/hidglobal.com/files/hid-understanding_card_data_formats-wp-en.pdf>

HID Corporation. "ThinLine II Reader." ThinLine® II Reader (n.d.): n. pag. ThinLine II Reader. HID Corporation. Web. 4 May 2015.

<http://www.hidglobal.com/sites/hidglobal.com/files/resource_files/prox-thinline-ii-reader-ds-en.pdf>