Computer Engineering Technology Department
CET 4711 – Computer Controlled System Design
Year & Semester: 2019 Spring
Instructor Name: Professor Farrukh Zia

Lab Report
Lab# and Title: Final Project – Attendance Tracker
Date: May 21, 2019

Student Name: Gene Nadela
Lab Partner Name: Ruposri Bhowmic

Lab# and Title: Final Project

1. Objective:

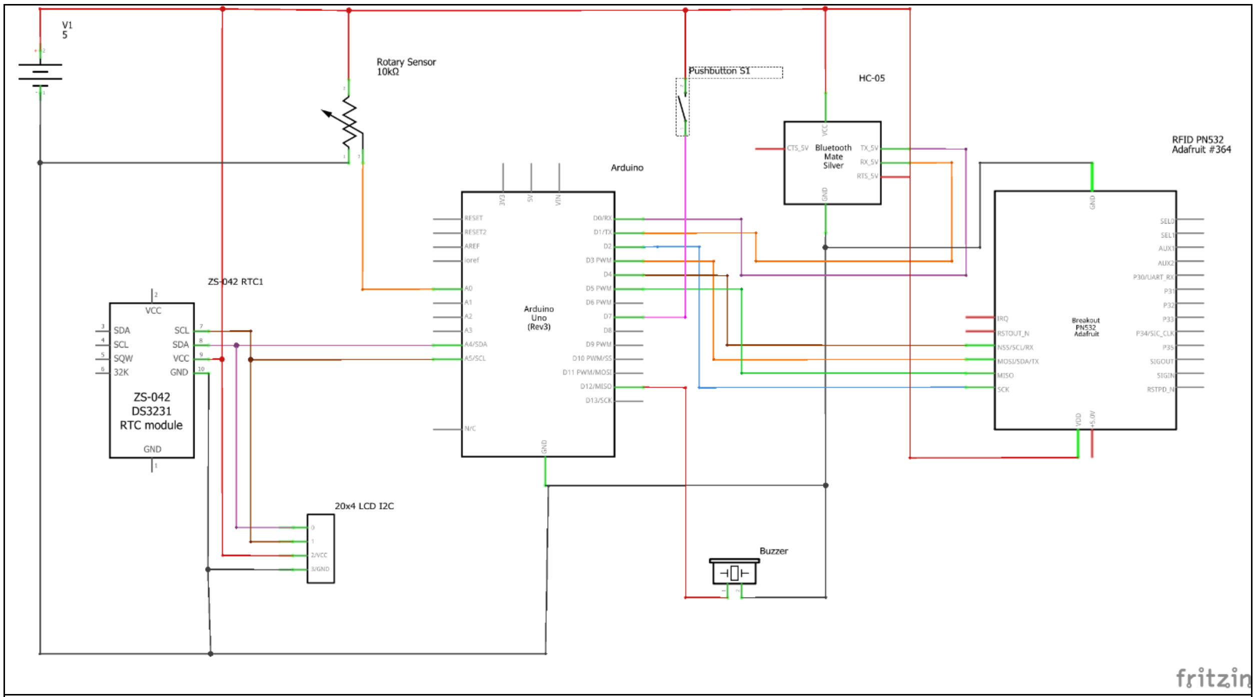
The purpose of this project is to develop a system that is able to track student attendance, using an RFID card as the primary means of identifying the student. The system is able to search its own records for the student, and assign to the student the time of their arrival, based on when the student has interacted with the device. It is able to notify the student of a successful read of the RFID card through sound and light indication. The student may also scroll through an LCD to see a list of each student for confirmation and time of arrival.

2. Procedure:

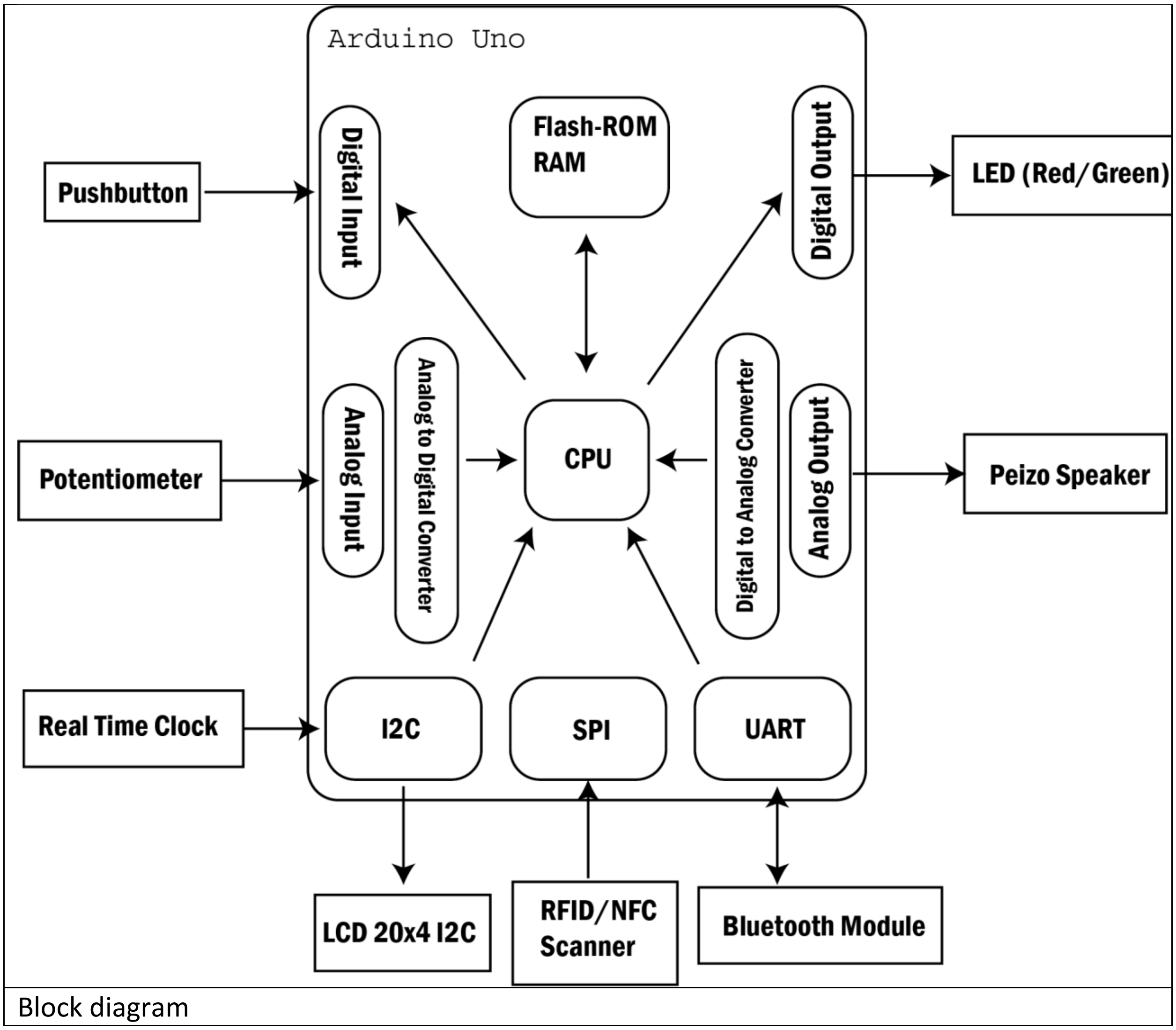
- Each input and output device is tested individually, and separately to determine the proper functionality of each.
- Starting with the LCD screen, its output is tested using dummy entries from an array.
- A potentiometer is placed, and tested for analog values. The minimum and maximum values are then used in mapping and constraining functions.
- The RFID is tested, using cards provided, and we determine each cards default 4 byte ID key, and record them for use in Student List Array.
- Piezo speaker is tested for different output tone values, and is used for audio confirmation for RFID reading.
- Real Time Clock is implemented, and library examined for functions to allow for Arduino to read hours, minutes values, then write to the Student List Array.
- Bluetooth is added, examined library for functions related to output. Output functions are used to transmit Student Array list data.
- Pushbutton added to allow for confirmation of Student Interaction via the LCD screen and potentiometer.
- Red/Green LED implemented for easy visual confirmation of RFID scan.

3. Diagrams:

- **Hardware:**

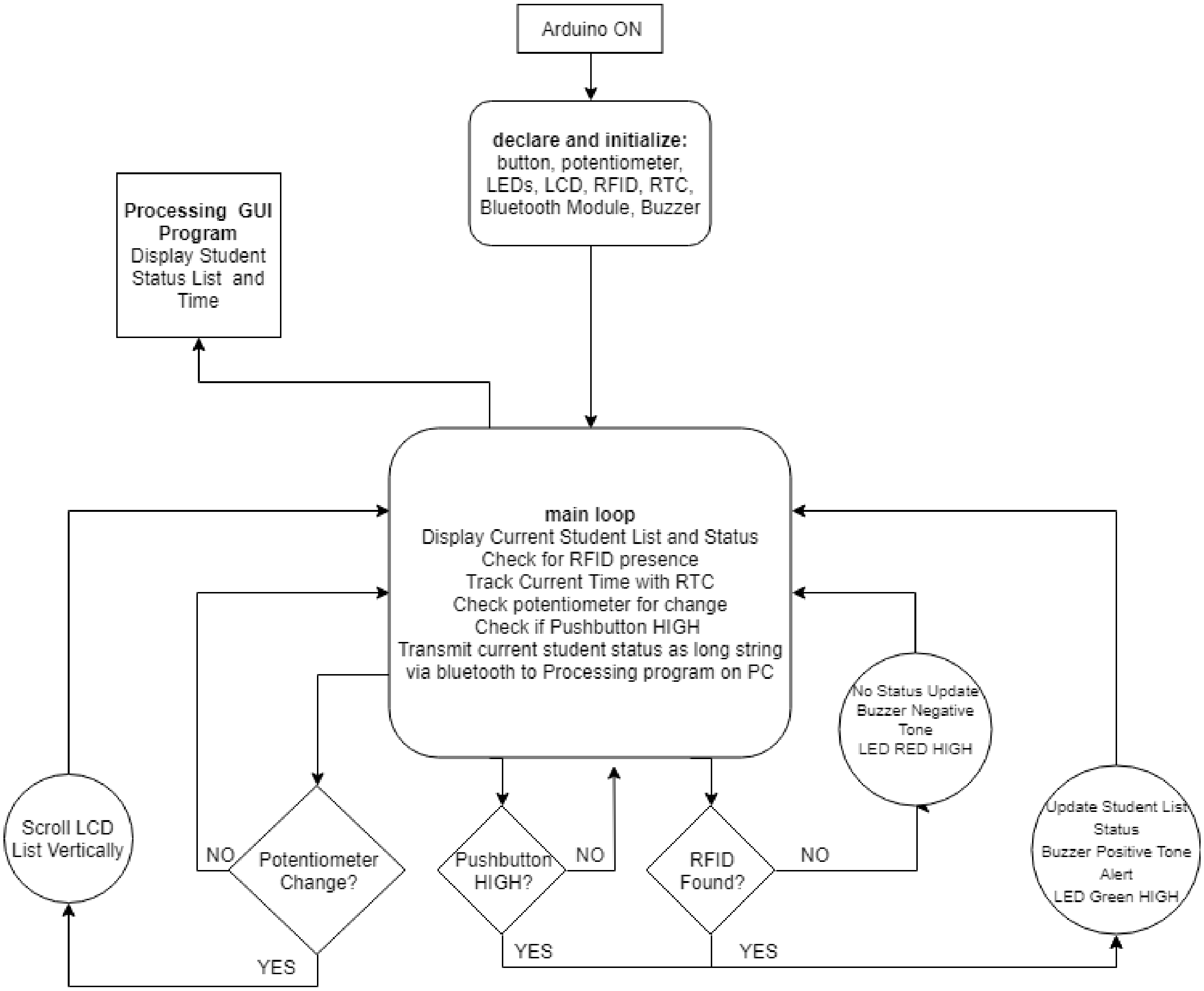


Schematic Diagram



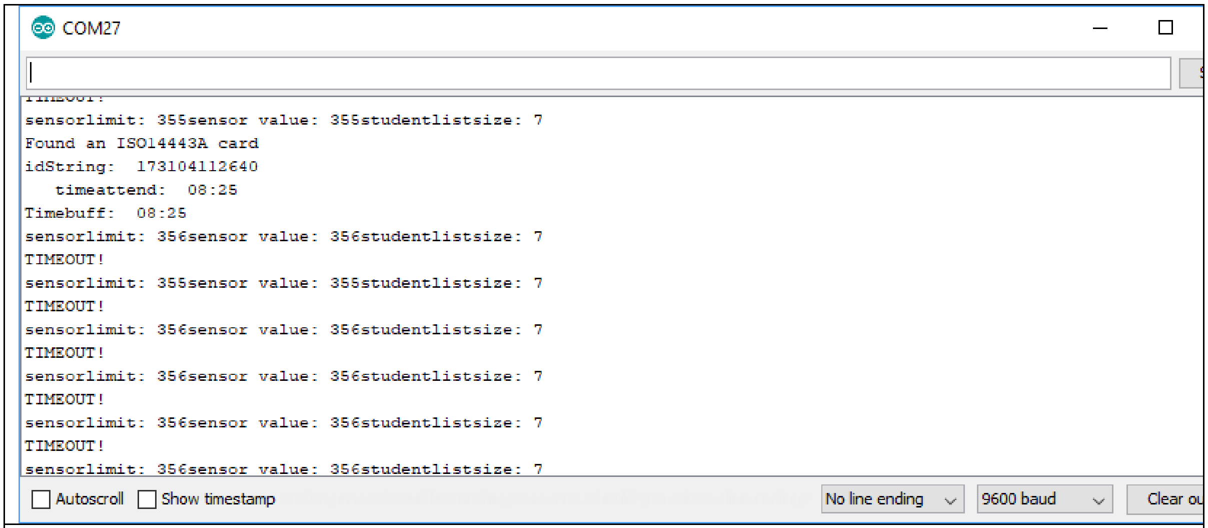
Block diagram

- **Software:**


Arduino ON
declare and initialize:
button, potentiometer,
LEDs, LCD, RFID, RTC,
Bluetooth Module, Buzzer
Processing GUI Program
Display Student Status List and Time
main loop
Display Current Student List and Status
Check for RFID presence
Track Current Time with RTC
Check potentiometer for change
Check if Pushbutton HIGH
Transmit current student status as long string via bluetooth to Processing program on PC
No Status Update
Buzzer Negative Tone
LED RED HIGH
Update Student List Status
Buzzer Positive Tone Alert
LED Green HIGH
Scroll LCD List Vertically
Potentiometer Change?
Pushbutton HIGH?
RFID Found?
NO
NO
NO
YES
YES
YES

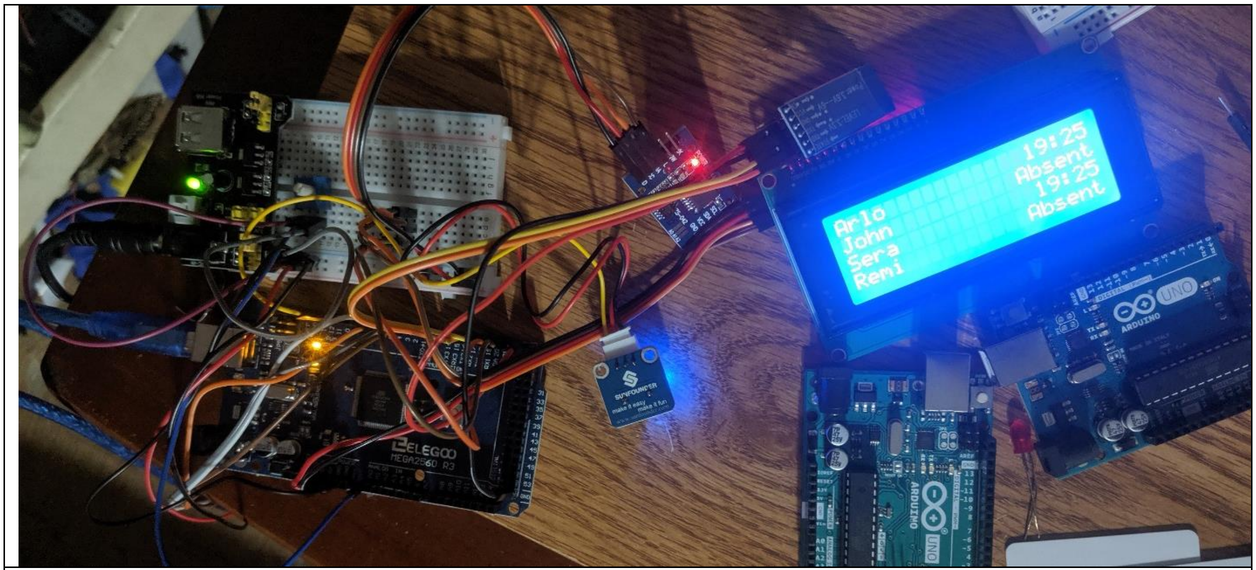
4. Measurements:

(Add captions to all screen shots, pictures and diagrams)

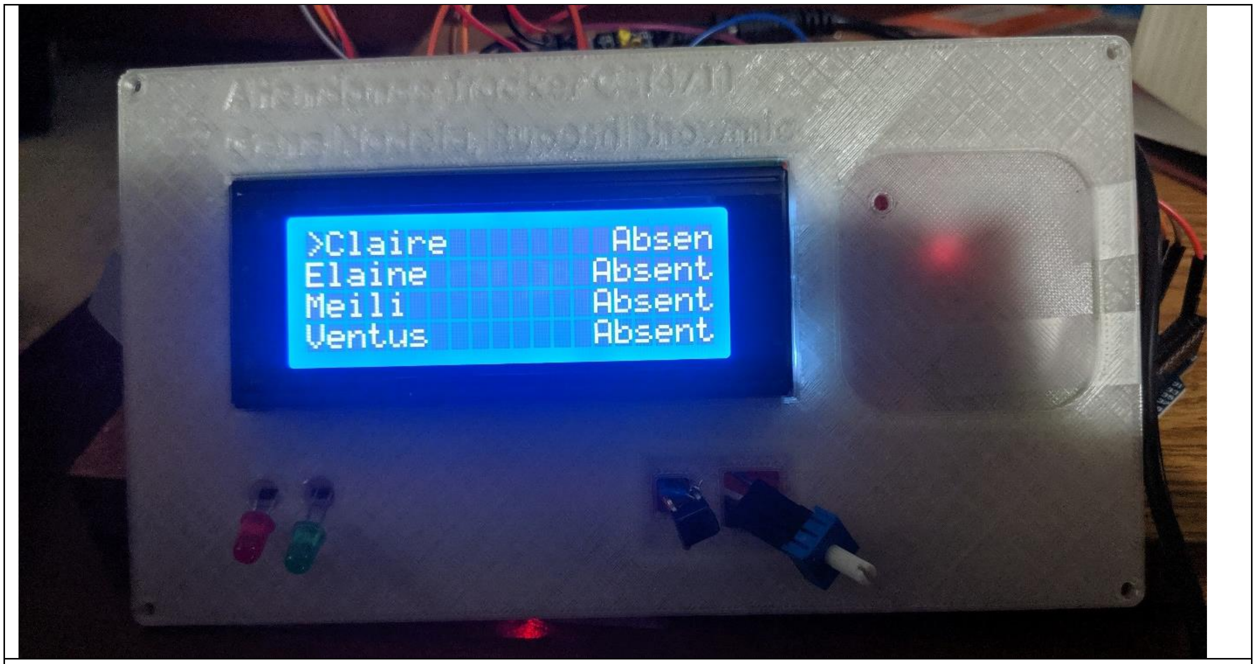


```
sensorlimit: 355sensor value: 355studentlistsize: 7
Found an ISO14443A card
idString:  173104112640
   timeattend:  08:25
Timebuff:  08:25
sensorlimit: 356sensor value: 356studentlistsize: 7
TIMEOUT!
sensorlimit: 355sensor value: 355studentlistsize: 7
TIMEOUT!
sensorlimit: 356sensor value: 356studentlistsize: 7
TIMEOUT!
sensorlimit: 356sensor value: 356studentlistsize: 7
TIMEOUT!
sensorlimit: 356sensor value: 356studentlistsize: 7
TIMEOUT!
sensorlimit: 356sensor value: 356studentlistsize: 7
```

Debugging through serial monitor. An ID card is indicated to have been read by the system. The system also continuously output the potentiometer sensor values.



Full test of setup with all components attached.



```
>Claire        Absen
Elaine        Absent
Meili         Absent
Ventus        Absent
```

Rough arrangement of final project.

5. Source code:

Arduino Code

```
/* Attendance Tracker
 *  This program stores a preset list of students and ID cards and their
status.
 *  It scans for RFID code, and dtermines if the student is present when the
student either
 *  places their ID onto the scan plate, or scrolls through a list on the
LCD and pushes a
 *  confirmation button. Confirmation results in an audio output, green LED
output, and updated entry on LCD, based on
 *  an RTC running time. Unconfirmed or failure, results in a negative
confirmation sounds and RED LED output.
 *  Gene Nadela and Ruposri Bhowmic
 *  May 20th, 2019
 *  v.1.0
 */

#include <SPI.h>
#include <Adafruit_PN532.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <RTClib.h>

#define ROTARY_ANGLE_SENSOR A0

#define PN532_SCK  (2)
#define PN532_MOSI (3)
#define PN532_SS   (4)
#define PN532_MISO (5)

LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 20
chars and 4 line display
Adafruit_PN532 cardscan(PN532_SCK, PN532_MISO, PN532_MOSI, PN532_SS);

RTC_DS1307 rtc;

int buzzerPin = 9;
int LIST_SIZE = 14;

int index;
int sensor_value;
int sensorlimit;
int StudentListSize;
int ledGREEN = 8;
int ledRED = 7;
int pushButton = 12;
int buttonState;

String timeAttend;
String timeAttendHour;
String timeAttendMinute;
char c;
char *classList[14][3]={ //Listed entries for class list
  {"Arlo", "Absent", "77157136640"},
  {"John", "Absent", "109117167640"},
  {"Sera", "Absent", "173104112640"},
  {"Remi", "Absent", "125174145640"},
  {"Blyke","Absent", "29247175640"},
  {"Isen", "Absent", "6127117640"},
  {"Cecille", "Absent", "XXXXXXX"},
  {"Claire", "Absent", "XXXXXXX"},
  {"Elaine", "Absent", "XXXXXXX"},
  {"Meili", "Absent", "XXXXXXX"},
  {"Ventus", "Absent", "XXXXXXX"},
  {"", "", ""},
  {"", "", ""},
  {"", "", ""},
};

void setup() {
  Serial.begin(9600); //Serial monitor setuo for debugging
  Serial1.begin(9600); //Serial1 for use with bluetooth connection

  // put your setup code here, to run once:
  lcd.init();  //initialize the lcd
  lcd.backlight();  //open the backlight
  cardscan.begin();
  cardscan.SAMConfig();
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledGREEN, OUTPUT);
  pinMode(ledRED, OUTPUT);
  pinMode(pushButton, INPUT);
  pinMode(ROTARY_ANGLE_SENSOR, INPUT);

  buttonState=0;
}

void loop() {

    uint8_t success;
    uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };  // Buffer to store the
returned UID
    uint8_t uidLength;
    String idString;

    DateTime now = rtc.now();
    char timebuff[6];
    timeAttend="";
    buttonState=digitalRead(pushButton);
    sensor_value = analogRead(ROTARY_ANGLE_SENSOR);
    sensorlimit = constrain(sensor_value, 0, 500);
    StudentListSize = map(sensorlimit, 0, 500, 0, LIST_SIZE-4);

    if(Serial1.available()){ //Checks forbluetooth status
      c=Serial1.read();
    }
  success=cardscan.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid,
&uidLength,100); //RFID library is used to check for card, return true for
success

  if(buttonState==HIGH) { //checks if pushbutton is HIGH, then updates
selected classList entry on screen with a new status
      buzzerAlert();
      timeAttendHour=AddLeadZero(now.hour());
      timeAttendMinute=AddLeadZero(now.minute());
      timeAttend.concat(timeAttendHour);
      timeAttend.concat(":");
      timeAttend.concat(timeAttendMinute);

      Serial.print("timeattend:  ");
      Serial.println(timeAttend);
      timeAttend.toCharArray(timebuff,6);
      classList[StudentListSize][1]=timebuff;
      Serial.print("Timebuff:  ");
      Serial.println(timebuff);
  }

  if (success) {
    // Prints ID card scan confirmation to Primary Serial Port
    Serial.println("Found an ISO14443A card");
    for (int k=0; k<5; k++) { //Form ID into String to be compared
        idString.concat(uid[k]);
      }
    Serial.print("idString:  ");
    Serial.println(idString);
    int studentRow=checkID(idString);

     if (idString==classList[studentRow][2]){ //compares the card key to the
assigned key for each student
        if(classList[studentRow][1]=="Absent") { //Check for Absent Status
          buzzerAlert();
          Serial.print("   ");
          timeAttendHour=AddLeadZero(now.hour());
          timeAttendMinute=AddLeadZero(now.minute());
          timeAttend.concat(timeAttendHour);
          timeAttend.concat(":");
          timeAttend.concat(timeAttendMinute);
          Serial.print("timeattend:  ");
          Serial.println(timeAttend);
          timeAttend.toCharArray(timebuff,6);
          classList[studentRow][1]=timebuff;
          Serial.print("Timebuff:  ");
          Serial.println(timebuff);
          idString="";
        }
        else {
          buzzerAlertNeg();
        }
      }


  }
```

```
DisplayStudentList();
SendListBluetooth();
}

//Determines if RFID is on list, and then returns the ID index
int checkID(String id) {

  for(int i=0; i<LIST_SIZE; i++) {
    if (id==classList[i][2]) {
    return i;
    }

  }
}

//Adds lead zero to string to prevent the summation of time integers on lcd
display
String AddLeadZero(int num) {
  String fullnum;
  if (num < 10) {
    fullnum = "0"+String(num);
  }
  else {
    fullnum=String(num);
  }
  return fullnum;
}

/*Prints out Student List and Status as one, long continuous string, and
places commas between entries
 * so that Processing IDE Software can split the entries into a vertical
list
 */
void SendListBluetooth() {
  for (int i=0; i<LIST_SIZE; i++) {
    Serial1.print(classList[i][0]);
    Serial1.print("   ");
    Serial1.print(classList[i][1]);
    Serial1.print(",");
  }
  Serial1.println();
}

//Buzzer and LED alert for when card is found to be on class list
void buzzerAlert() {
  digitalWrite(ledGREEN, HIGH);
  tone(buzzerPin,1000);
  delay(100);
  noTone(buzzerPin);
  delay(50);
  tone(buzzerPin,2000);
  delay(100);
  noTone(buzzerPin);
  delay(1000);
  digitalWrite(ledGREEN, LOW);
}
```

```
//Buzzer and LED alert for when card is found to not be on class list
void buzzerAlertNeg() {
  digitalWrite(ledRED, HIGH);
  tone(buzzerPin,500);
  delay(100);
  noTone(buzzerPin);
  delay(50);
  tone(buzzerPin,250);
  delay(100);
  noTone(buzzerPin);
  delay(1000);
  digitalWrite(ledRED, LOW);
}

//Allows scrolling of entries on LCD display using potentiometer
void DisplayStudentList() {
    lcd.clear(); //neaded to clear characters from screen
    lcd.setCursor(0,0);
    lcd.print(">");
    lcd.print(classList[StudentListSize][0]);
    getSpacing(classList[StudentListSize][0],classList[StudentListSize][1]);
    lcd.print(classList[StudentListSize][1]);
    lcd.setCursor(0, 1);
    lcd.print(classList[StudentListSize+1][0]);

getSpacing(classList[StudentListSize+1][0],classList[StudentListSize+1][1]);
    lcd.print(classList[StudentListSize+1][1]);
    lcd.setCursor(0, 2);
    lcd.print(classList[StudentListSize+2][0]);

getSpacing(classList[StudentListSize+2][0],classList[StudentListSize+2][1]);
    lcd.print(classList[StudentListSize+2][1]);
    lcd.setCursor(0, 3);
    lcd.print(classList[StudentListSize+3][0]);

getSpacing(classList[StudentListSize+3][0],classList[StudentListSize+3][1]);
    lcd.print(classList[StudentListSize+3][1]);
    //delay(100);
}

//adds a static amount of spacing between name and status entry on LCD
void getSpacing(String studentName, String studentStatus){
    for (int i=0; i<(20-studentStatus.length()-studentName.length()); i++) {
      lcd.print(" ");
    }
}
```

Processing IDE Code

```
/*
Attendence Tracker GUI

This program processes the string received from the attendance tracker
device via bluetooth connection.
It is split, and put into a vertical list of entries.

Gene Nadela and Rupsori Bhowmic
May 20th, 2019
v.1.0

*/


import processing.serial.*;
import controlP5.*;
Serial myPort;
String studentStatus="";
ControlP5 cp5;
String[] classList;


void setup(){
  size(500, 900);
  myPort = new Serial(this, "COM29", 9600); // Starts the serial
communication
  myPort.bufferUntil('\n'); // Defines up to which character the data from
the serial port will be read. The character '\n' or 'New Line'
  cp5 = new ControlP5(this);
}

void serialEvent(Serial myPort){

  studentStatus = myPort.readStringUntil('\n');

}

void draw() {
  classList = split(studentStatus, ','); //Splits received string from
bluetooth into vertical list of entries
  background(0,0,0);
  textSize(18);
  for (int i=0; i<11; i++) {
    text(classList[i], 100, 100+i*30);
  }



}
```

6. Troubleshooting:

- **Symptoms of the problem:**
  RFID Breakout Not Recognized by Arduino
- **Hardware and/or software test performed to determine the cause of the problem**
  Examined the breakout pins physically, determined not soldered correctly.
- **Solution to fix the problem:**
  Desoldered the breakout pins, and soldered in a new set.

- **Symptoms of the problem:**
  LCD not refreshing display with new information. Text entries overlap with one another.
- **Hardware and/or software test performed to determine the cause of the problem**
  Examined software, and looked for differences in sample code.
- **Solution to fix the problem:**
  Determined that specific functions had different naming schemes that conflicted with current library. Examined header files for the correct function, and used lcd.clear() function to clear and refresh the LCD output screen during void loop().

- **Symptoms of the problem:**
  System crash when potentiometer is at highest value
- **Hardware and/or software test performed to determine the cause of the problem**
  Checked different potentiometers to rule out hardware issues. Examined code of value index and their entries as sensor value increased.
- **Solution to fix the problem:**
  Determined that when system attempts to recall character array out of bounds, the system crashes. Add blank entries to student list to allow the LCD to scroll into blanks, prevented crashes.

7. Discussion:

Input sub-system:

Physical sensor(s) used: Pushbutton, Potentiometer, RFID/NFC Scanner, Real Time Clock
Interface circuit:
Pushbutton – Pin 12 - Digital Input, to determine HIGH and LOW state.
Potentiometer – Pin A0 - Analog Input, ADC to determine sensor value.
Real Time Clock – I2C – ongoing clock data to determine the time of interaction.
RFID/NFC Scanner – SPI – For continuous scanning for ID cards.

Data Input operation:

Pushbutton:

```
if(buttonState==HIGH)
```

Potentiometer:

```
sensor_value = analogRead(ROTARY_ANGLE_SENSOR);
```

Real Time Clock:

```
DateTime now = rtc.now();
```

RFID/NFC Scanner::

```
success=cardscan.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid,
&uidLength,100)
```

Data Processing sub-system:

Micro-controller:

ATmega328P on Arduino board, 16 MHz clock, RISC architecture

Data Processing operation:

(Convert RTC Values to String, Placed into Student List Array)

```
    timeAttendHour=AddLeadZero(now.hour());
    timeAttendMinute=AddLeadZero(now.minute());
    timeAttend.concat(timeAttendHour);
    timeAttend.concat(":");
    timeAttend.concat(timeAttendMinute);
    timeAttend.toCharArray(timebuff,6);
       classList[StudentListSize][1]=timebuff;
```

(Read sensor values from potentiometer, constrain to specified sensor readings, then map to value of indices based on the size of the Student list)

```
    sensor_value = analogRead(ROTARY_ANGLE_SENSOR);
    sensorlimit = constrain(sensor_value, 0, 500);
    StudentListSize = map(sensorlimit, 0, 500, 0, LIST_SIZE-4);
```

Data Storage operation:

```
char *classList[14][3]={ //Listed entries for class list
  {"Arlo", "Absent", "77157136640"},
  {"John", "Absent", "109117167640"},
  {"Sera", "Absent", "173104112640"},
  {"Remi", "Absent", "125174145640"},
  {"Blyke","Absent", "29247175640"},
  {"Isen", "Absent", "6127117640"},
  {"Cecille", "Absent", "XXXXXXX"},
  {"Claire", "Absent", "XXXXXXX"},
  {"Elaine", "Absent", "XXXXXXX"},
  {"Meili", "Absent", "XXXXXXX"},
  {"Ventus", "Absent", "XXXXXXX"},
  {"", "", ""},
  {"", "", ""},
  {"", "", ""},
   };
```

Program

Size of **FLASH memory** used for storing program code (kB):	13.170 kB
Size of **RAM** used for storing variables (kB):	1.065 kB

Output sub-system:

Output device(s) used: Buzzer/Peizo Speaker, LED(Red/Green), LCD 20x4

Interface circuit:

Buzzer – Pin 9 – Analog Output Sound confirmation

LED(Red/Green) – Pin 7 and 8, respectively. Digital Output confirmation.

LCD 20x4 – I2C – Digital Output for text lists.

Data Output operation:

(Buzzer/LED Output Function):

```
void buzzerAlert() {
  digitalWrite(ledGREEN, HIGH);
  tone(buzzerPin,1000);
  delay(100);
  noTone(buzzerPin);
  delay(50);
  tone(buzzerPin,2000);
  delay(100);
  noTone(buzzerPin);
  delay(1000);
  digitalWrite(ledGREEN, LOW);
}
void buzzerAlertNeg() {
  digitalWrite(ledRED, HIGH);
  tone(buzzerPin,500);
  delay(100);
  noTone(buzzerPin);
  delay(50);
  tone(buzzerPin,250);
  delay(100);
  noTone(buzzerPin);
  delay(1000);
  digitalWrite(ledRED, LOW);
   }
```

(LCD 20x4):

```
void DisplayStudentList() {
    lcd.clear(); //neaded to clear characters from screen
    lcd.setCursor(0,0);
    lcd.print(">");
    lcd.print(classList[StudentListSize][0]);

getSpacing(classList[StudentListSize][0],classList[StudentListSize][1]);
    lcd.print(classList[StudentListSize][1]);
    lcd.setCursor(0, 1);
    lcd.print(classList[StudentListSize+1][0]);

getSpacing(classList[StudentListSize+1][0],classList[StudentListSize+1][1]
);
    lcd.print(classList[StudentListSize+1][1]);
    lcd.setCursor(0, 2);
    lcd.print(classList[StudentListSize+2][0]);

getSpacing(classList[StudentListSize+2][0],classList[StudentListSize+2][1]
);
    lcd.print(classList[StudentListSize+2][1]);
    lcd.setCursor(0, 3);
    lcd.print(classList[StudentListSize+3][0]);

getSpacing(classList[StudentListSize+3][0],classList[StudentListSize+3][1]
);
    lcd.print(classList[StudentListSize+3][1]);
    //delay(100);
   }
```

Data Communication sub-system:

Communication interface used:

Serial Communication via USB with PC.

Serial Communication via Bluetooth to PC.

Data Communication operation:

```
Serial.begin(9600); //Serial monitor setuo for debugging
  Serial1.begin(9600); //Serial1 for use with bluetooth connection

void SendListBluetooth() {
  for (int i=0; i<LIST_SIZE; i++) {
    Serial1.print(classList[i][0]);
    Serial1.print("   ");
    Serial1.print(classList[i][1]);
    Serial1.print(",");
  }
  Serial1.println();
   }
```

8. Conclusion:

The Attendance Tracker makes an attempt to maximize the use of known subsystem available to the Arduino microcontroller. Digital and Analog inputs are covered by the Pushbutton and Potentiometer respectively, as well as the RFID scanner and RTC module. Digital and Analog output are covered by the LED and Peizo Speaker, respectively, as well as the LCD module. Data communication uses both Serial for debugging purposes, and Bluetooth for actual remote transmission of data. Implementation of all components proved especially challenging.

Practical Use

Keeping tracker of the presence of students and arrival time can be a tedious task. This purpose of this device is to streamline this process so as to minimize the time to perform this activity. It also introduces a greater degree of accuracy should the teacher, at their discretion, can mark a student late. This system could also find use in workplace environments, where keeping track of employee work schedule may be important in determining payroll.