Salt Dispensing RC Car

Team Members:

Omar Salas-Rodriguez	osalas3	osalas3@uic.edu
Harsh Gupta	hgupta20	hgupta20@uic.edu
Brian Goldenberg	bgolde2	bgolde2@uic.edu

Abstract:

Using a mobile application we will be designing a car that will maneuver through ice and dispense salt on the ground creating a safer walking environment. With three arduinos, a motor controller for the dual motors, chassis with custom 3D printed parts, wheels, chain, stepper motor and bluetooth module will be will dispense salt from the back of the tank. When in need of salt, the tank will pull up under a nearby base to request more salt to be dispensed(only when the car is in the proper position). All controlled from the mobile application that we will also design and implement.

Description:

The Salt Dispensing RC Car has one major task to perform, allowing the user to control the RC Car through a companion mobile application and dispense salt on the icy ground. The mobile app will allow the user to dispense the salt from the car to the ground, but it will also allow them to dispense more salt from the base onto the car. The base containing the extra salt will have a spot underneath for the car to get into place to receive some more salt. The application will have a button that will dispense the salt onto the car which will only be clickable when the car is in its proper spot. With the mobile app and the base being full of salt, the RC car will be able to maneuver all throughout the icy ground and dispense the salt so that the ground is iceless.

- **RC Car -** The car will communicate with the base station and the user and dispense salt according to the need. The RC would have one Arduino onboard which will control the car as well as the tank having salt which would use a servo.
- **Base Station**- It is basically a refilling station where the tank would arrive after it runs out of salt. The station would have one Arduino which would control the sensors to make sure the RC car is beneath it and would then add salt to the tank. The station will be using ultrasonic sensors to sense the proximity of the car.

Input Devices:

Distance sensing- Distance Sensing will be used to measure if the car is at the proper salt dispensing location.
Mobile application - The mobile application will be used to maneuver the car. The app will have a button to dispense salt from the car to the ground and a seperate button to dispense salt from the base to the car.
Bluetooth Module - Used to send signals to the car from the mobile app. The bluetooth signal will also send messages to the mobile app as to whether the car is at the proper loading zone or not.
Ultra Sonic sensors and Stepper motors

Output Devices:

3 Motor control - A total of three motor controllers will need to be used. The first will be used to maneuver the car. The second motor controller will be used to manage the valve and control the salt dispensed from the car. The third motor controller will manage the valve on the base and control the salt being dispensed to the car.
Stepper Motor and bunch of leds.

Arduino Communication:

With the utilization of 3 arduinos, there will be lots of communication amongst all of them. The arduino controlling the car will be the main arduino that will contain the bluetooth module which will communicate with the companion app for controlling the motors. The same arduino will be communicating with the second arduino on the car to see if the salt needs to be dispensed to the ground. The central arduino will also have to communicate with the arduino controlling the base so that the base arduino knows when it's in the correct location. When the car is in the spot, the arduino communicates back to the central one so that the mobile application allows the user to press the dispense button to refill the container of salt on the car.

Timeline:

Week 7: Develop code for arduinos and app

- Code developed for the RC Car
- RC car debugging

Week 8: Build Chassis, wheels and chains,

- Chasis built and the car is operatable
- 3D printing the Salt Tank

Week 9: Assemble all the parts of the RC car and implement the code

- Writing the code for the servo motor to dispense salt
- 3D printing the tank because of issues
- Rebuilding the chassis and making some changes

Week 10: Implement the code of RC car and the dispenser

- Yet to implement the code for the tank
- Code for the RC has been implemented
- Communication between the arduinos and bluetooth

Week 11: Testing and debugging

- Need to implement the code for the salt tank
- Need to implement the code for the Base Station

Week 12: Testing and debugging

- Had to redo the servo code as not enough power to turn the spiral

Week 13: Debugging major issues with Salt Dispenser

- Used a stepper motor to turn the spiral in the dispenser to 360 degree.

Week 14: Demo and Presentation

Materials:

The car will require several important parts. First, it will be built with a 3D printed chassis, tank wheels, and chains that fit to perfection. Then, it will require a motor controller to control and power 2 DC motors for the wheels. The motor controller will be connected to an external battery pack for power and will be connected to a central arduino that will control the speed and direction of the wheels. The car will also have a small container that will hold the salt which will be controlled by a second arduino. The tank will have a handle to dispense or a valve that will dispense the salt which will be controlled by the arduino. Lastly, the car will need a bluetooth module so that the arduino is able to communicate with the companion app and vise versa.

The base with the salt will have a large container to hold the salt and will have a valve underneath for the salt to be dispensed. This base will require another arduino which will control the valve. The valve will open when the car is underneath the base using the companion app.
We will also need an LCD and some 3D printed parts for tanks.

References:

Arduino motor controller

pfodApp (Android) Bluetooth, BLE, SMS and WiFi. Works with 101 and other BLE boards. No Android Programming Required. pfodDesigner generates Arduino code for 101 and other BLE boards and WiFi and SMS shields. No Android Programming Required
Annikken Andee Don't need iOS or Android programming to create functional mobile interface, buttons, sliders, graphs, text input and many more for customisation.
New ping - distance sensing

Original Work:

We will be using a guide on creating an RC Car, but we will be incorporating the tank design and container. The bluetooth companion app will be modified from a simple data sending and receiving guide to support the controls of the motor and the dispenser communication. The idea for the base containing the salt is unique to the project and will incorporate simple components that play a crucial part to the project.

Code:

Arduino Tank code (WIP):

```
#define E1 6
#define DIRA1 7
#define DIRA2 8

#define DIRB2 4
#define DIRB1 3
#define E2 5

char command;
int tank_speed;
String str;

void setup() {
  pinMode(E1, OUTPUT);
  pinMode(DIRA1, OUTPUT);
  pinMode(DIRA2, OUTPUT);
  pinMode(DIRB2, OUTPUT);
  pinMode(DIRB1, OUTPUT);
  pinMode(E2, OUTPUT);

  Serial.begin(9600);
  tank_speed = 180;


}

void goForwards(){
  analogWrite(E1,tank_speed);
  digitalWrite(DIRA1,LOW);
  digitalWrite(DIRA2,HIGH);

  analogWrite(E2,tank_speed);
  digitalWrite(DIRB1,LOW);
  digitalWrite(DIRB2,HIGH);
}
void goBackwards(){
  analogWrite(E1,tank_speed);
  digitalWrite(DIRA1,HIGH);
  digitalWrite(DIRA2,LOW);

  analogWrite(E2,tank_speed);
  digitalWrite(DIRB1,HIGH);
  digitalWrite(DIRB2,LOW);
}
void goRight(){
  analogWrite(E1,tank_speed);
  digitalWrite(DIRA1,HIGH);
  digitalWrite(DIRA2,LOW);

  analogWrite(E2,tank_speed);
  digitalWrite(DIRB1,LOW);
  digitalWrite(DIRB2,HIGH);
}
void goLeft(){
  analogWrite(E1,tank_speed);
  digitalWrite(DIRA1,LOW);
  digitalWrite(DIRA2,HIGH);

  analogWrite(E2,tank_speed);
  digitalWrite(DIRB1,HIGH);
  digitalWrite(DIRB2,LOW);
}
void stopTank(){
  digitalWrite(DIRA1,LOW);
  digitalWrite(DIRA2,LOW);
  digitalWrite(DIRB1,LOW);
  digitalWrite(DIRB2,LOW);
}

void receiveMessage(){
  if (Serial.available() > 0)
    str = "";
  while(Serial.available() > 0){
    command = ((byte)Serial.read());
    if(command == ':'){
      break;
    }else{
      str += command;
    }
    delay(1);
  }
  Serial.println(str);
  if(str == "FORWARD"){
    goForwards();
  }else if(str == "RIGHT"){
    goRight();
  }else if(str == "REVERSE"){
    goBackwards();
  }else if(str == "LEFT"){
    goLeft();
  }else if(str == "STOP"){
    stopTank();
  }else if(str == "GET"){
    getSalt();
  }else if(str == "DROP"){
    dropSalt();
  }else if(str.substring(0,5) == "SPEED"){
    tank_speed = str.substring(5, str.length()).toInt();
  }
}

void loop() {
  receiveMessage();
}
```

```
177  void motorStep(){
178    switch(stepNum){ // Turn each piece of the stepper motor
179      case 0:
180        digitalWrite(STEPPER_PIN_1, HIGH);
181        digitalWrite(STEPPER_PIN_2, LOW);
182        digitalWrite(STEPPER_PIN_3, LOW);
183        digitalWrite(STEPPER_PIN_4, LOW);
184        break;
185      case 1:
186        digitalWrite(STEPPER_PIN_1, LOW);
187        digitalWrite(STEPPER_PIN_2, HIGH);
188        digitalWrite(STEPPER_PIN_3, LOW);
189        digitalWrite(STEPPER_PIN_4, LOW);
190        break;
191      case 2:
192        digitalWrite(STEPPER_PIN_1, LOW);
193        digitalWrite(STEPPER_PIN_2, LOW);
194        digitalWrite(STEPPER_PIN_3, HIGH);
195        digitalWrite(STEPPER_PIN_4, LOW);
196        break;
197      case 3:
198        digitalWrite(STEPPER_PIN_1, LOW);
199        digitalWrite(STEPPER_PIN_2, LOW);
200        digitalWrite(STEPPER_PIN_3, LOW);
201        digitalWrite(STEPPER_PIN_4, HIGH);
202        break;
203    }
204    // Make stepper motor keep on turning
205    stepNum++;
206    if(stepNum > 3)
207      stepNum = 0;
208  }
209
210  void loop() {
211    receiveMessage();
212
213    if(drop)
214      motorStep(); // Start stepper in reverse motion
215    delay(2);
```

Android Companion App:

Pairing Activity Class:

```
17  public class PairingActivity extends AppCompatActivity {
18
19      private ListView bluetoothDevices;
20      private BluetoothAdapter bluetoothAdapter = null;
21      private ArrayList<String> addresses = new ArrayList<>();
22      public static String EXTRA_ADDRESS = "DEVICE_ADDRESS";
23
24      @Override
25      protected void onCreate(Bundle savedInstanceState) {
26          super.onCreate(savedInstanceState);
27          setContentView(R.layout.activity_main);
28          bluetoothDevices = findViewById(R.id.device_list);
29          bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
30          if(bluetoothAdapter == null){
31              Toast.makeText(getApplicationContext(), "Bluetooth Device Not Available", Toast.LENGTH_LONG).show();
32              finish();
33          }
34          else if(!bluetoothAdapter.isEnabled()){
35              //Ask to the user turn the bluetooth on
36              Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
37              startActivityForResult(turnBTon,1);
38          }
39      }
```

```
41      @Override
42      protected void onResume() {
43          super.onResume();
44          Set<BluetoothDevice> pairedDevices = bluetoothAdapter.getBondedDevices();
45          ArrayList<String> list = new ArrayList<>();
46
47          if (pairedDevices.size() > 0)
48              for (BluetoothDevice bt : pairedDevices) {
49                  list.add(bt.getName() + "\n" + bt.getAddress()); //Get the device's name and the address
50                  addresses.add(bt.getAddress());
51              }
52          else
53              Toast.makeText(getApplicationContext(), "No Paired Bluetooth Devices Found.", Toast.LENGTH_LONG).show();
54
55          bluetoothDevices.setAdapter(new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, list));
56          bluetoothDevices.setOnItemClickListener(new AdapterView.OnItemClickListener() {
57              @Override
58              public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
59                  Intent i = new Intent(PairingActivity.this, ControllerActivity.class);
60                  i.putExtra(EXTRA_ADDRESS, addresses.get(position));
61                  startActivity(i);
62              }
63          });
64      }
65  }
```

Controller Activity:

```
22  public class ControllerActivity extends AppCompatActivity {
23
24      private int tankSpeed;
25      private String address;
26      private ProgressDialog progress;
27      private BluetoothSocket btSocket = null;
28      private boolean isBtConnected = false;
29      private static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
30
31      @SuppressLint("ClickableViewAccessibility")
32      @Override
33      protected void onCreate(Bundle savedInstanceState) {
34          super.onCreate(savedInstanceState);
35          setContentView(R.layout.activity_controller);
36
37          Intent i = getIntent();
38          address = i.getStringExtra(PairingActivity.EXTRA_ADDRESS);
39          new ConnectBT().execute();
40
41          Button btnDropSalt = findViewById(R.id.drop_salt);
42          Button btnGetSalt = findViewById(R.id.get_salt);
43          Button btnForward = findViewById(R.id.btnForward);
44          Button btnReverse = findViewById(R.id.btnReverse);
45          Button btnLeft = findViewById(R.id.btnLeft);
46          Button btnRight = findViewById(R.id.btnRight);
47          Button btnDisc = findViewById(R.id.btnDisc);
48          SeekBar tank_speed_bar = findViewById(R.id.tank_speed_bar);
49          tankSpeed = 0;
50
```

```
51              btnDropSalt.setOnTouchListener(new View.OnTouchListener() {
52                  @Override
53                  public boolean onTouch(View v, MotionEvent event) {
54                      int action =  event.getAction();
55                      if(action == MotionEvent.ACTION_DOWN)
56                          dropSalt();
57                      else if(action == MotionEvent.ACTION_UP)
58                          moveStop();
59                      return false;
60                  }
61              });
62
63              btnGetSalt.setOnTouchListener(new View.OnTouchListener() {
64                  @Override
65                  public boolean onTouch(View v, MotionEvent event) {
66                      int action =  event.getAction();
67                      if(action == MotionEvent.ACTION_DOWN)
68                          getSalt();
69                      else if(action == MotionEvent.ACTION_UP)
70                          moveStop();
71                      return false;
72                  }
73              });
74
75              btnForward.setOnTouchListener(new View.OnTouchListener(){
76                  @Override
77                  public boolean onTouch(View v, MotionEvent event){
78                      int action =  event.getAction();
79                      if(action == MotionEvent.ACTION_DOWN)
80                          moveForward();
81                      else if(action == MotionEvent.ACTION_UP)
82                          moveStop();
83                      return false;
84                  }
85              });
```

```
86
87          btnReverse.setOnTouchListener(new View.OnTouchListener(){
88              @Override
89              public boolean onTouch(View v, MotionEvent event){
90                  int action =  event.getAction();
91                  if(action == MotionEvent.ACTION_DOWN)
92                      moveReverse();
93                  else if(action == MotionEvent.ACTION_UP)
94                      moveStop();
95                  return false;
96              }
97          });
98
99          btnLeft.setOnTouchListener(new View.OnTouchListener(){
100             @Override
101             public boolean onTouch(View v, MotionEvent event){
102                 int action =  event.getAction();
103                 if(action == MotionEvent.ACTION_DOWN)
104                     moveLeft();
105                 else if(action == MotionEvent.ACTION_UP)
106                     moveStop();
107                 return false;
108             }
109         });
110
111         btnRight.setOnTouchListener(new View.OnTouchListener(){
112             @Override
113             public boolean onTouch(View v, MotionEvent event){
114                 int action =  event.getAction();
115                 if(action == MotionEvent.ACTION_DOWN)
116                     moveRight();
117                 else if(action == MotionEvent.ACTION_UP)
118                     moveStop();
119                 return false;
120             }
121         });
```

```
124         btnDisc.setOnClickListener(new View.OnClickListener(){
125             @Override
126             public void onClick(View v){
127                 Disconnect(); //close connection
128             }
129         });
130 
131         tank_speed_bar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
132             @Override
133             public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
134                 tankSpeed = progress + 180;
135             }
136 
137             @Override
138             public void onStartTrackingTouch(SeekBar seekBar) {
139 
140             }
141 
142             @Override
143             public void onStopTrackingTouch(SeekBar seekBar) {
144                 updateSpeed();
145             }
146         });
147 
148     }
149 
150     private void getSalt(){
151         if (btSocket!=null){
152             try{
153                 btSocket.getOutputStream().write("DROP".getBytes());
154             }catch (IOException e){
155                 Log.e("DROP: ", e.toString());
156             }
157         }
158     }
```

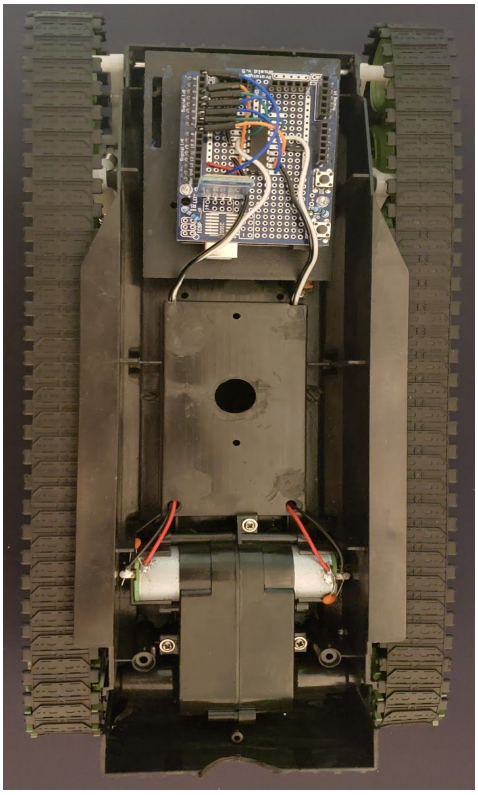
```
160     private void dropSalt(){
161         if (btSocket!=null){
162             try{
163                 btSocket.getOutputStream().write("GET".getBytes());
164             }catch (IOException e){
165                 Log.e("GET: ", e.toString());
166             }
167         }
168     }
169
170     private void updateSpeed(){
171         if(btSocket != null){
172             try{
173                 btSocket.getOutputStream().write(("SPEED" + tankSpeed).getBytes());
174             }catch(IOException e){
175                 Log.e("TANK SPEED: ", e.toString());
176             }
177         }
178     }
179
180     private void Disconnect(){
181         if (btSocket!=null){ //If the btSocket is busy
182             try{
183                 btSocket.close(); //close connection
184             }catch (IOException e){
185                 Log.e("DISCONNECT: ", e.toString());
186             }
187         }
188         finish(); //return to the first layout
189
190     }
```

```
191
192     private void moveForward(){
193         if (btSocket!=null){
194             try{
195                 btSocket.getOutputStream().write("FORWARD".getBytes());
196             }catch (IOException e){
197                 Log.e("FORWARD: ", e.toString());
198             }
199         }
200     }
201
202     private void moveReverse(){
203         if (btSocket!=null){
204             try{
205                 btSocket.getOutputStream().write("REVERSE".getBytes());
206             }catch (IOException e){
207                 Log.e("REVERSE: ", e.toString());
208             }
209         }
210     }
211
212     private void moveLeft(){
213         if (btSocket!=null){
214             try{
215                 btSocket.getOutputStream().write("LEFT".getBytes());
216             }catch (IOException e){
217                 Log.e("LEFT: ", e.toString());
218             }
219         }
220     }
221
```

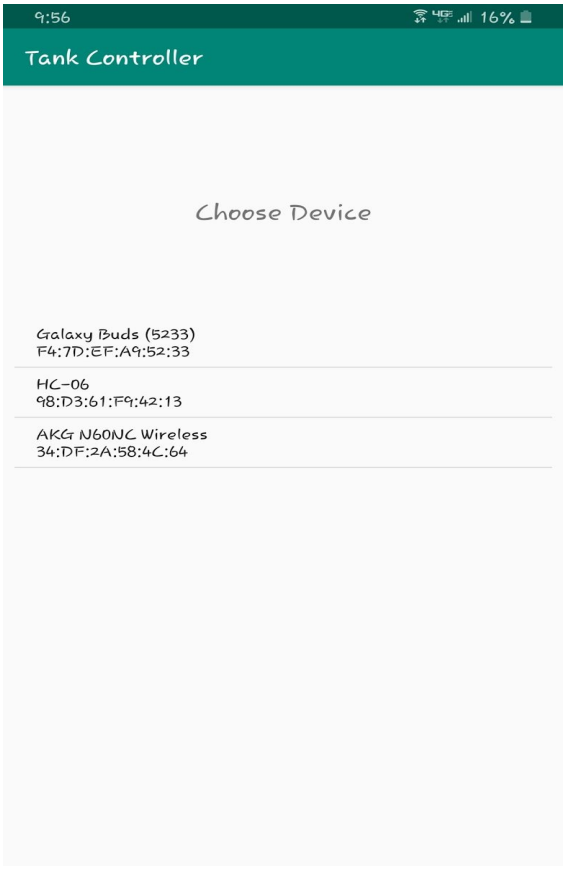
```
    private void moveRight(){
        if (btSocket!=null){
            try{
                btSocket.getOutputStream().write("RIGHT".getBytes());
            }catch (IOException e){
                Log.e("RIGHT: ", e.toString());
            }
        }
    }

    private void moveStop(){
        if (btSocket!=null){
            try{
                btSocket.getOutputStream().write("STOP".getBytes());
            }catch (IOException e){
                Log.e("STOP: ", e.toString());
            }
        }
    }

    private class ConnectBT extends AsyncTask<Void, Void, Void> { //Bluetooth Thread
        private boolean ConnectSuccess = true; //if it's here, it's almost connected

        @Override
        protected void onPreExecute(){
            progress = ProgressDialog.show(ControllerActivity.this, "Connecting...", "Please wait!!!");  //show a progress dialog
        }


       @Override
       protected Void doInBackground(Void... devices){ //while the progress dialog is shown, the connection is done in background
           try {
               if (btSocket == null || !isBtConnected){
                   BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();
                   BluetoothDevice dev = btAdapter.getRemoteDevice(address);//connects to the device's address and checks if it's available
                   btSocket = dev.createInsecureRfcommSocketToServiceRecord(myUUID);//create a RFCOMM (SPP) connection
                   BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                   btSocket.connect();//start connection
               }
           }
           catch (IOException e){
               ConnectSuccess = false;//if the try failed, you can check the exception here
           }
           return null;
       }
       @Override
       protected void onPostExecute(Void result){ //after the doInBackground, it checks if everything went fine
           super.onPostExecute(result);

           if (!ConnectSuccess) {
               Toast.makeText(getApplicationContext(),"Connection Failed. Is it a SPP Bluetooth? Try again.",Toast.LENGTH_LONG).show();
               finish();
           }else{
               Toast.makeText(getApplicationContext(),"Connected.",Toast.LENGTH_LONG).show();
               isBtConnected = true;
           }
           progress.dismiss();
       }
    }

}
```

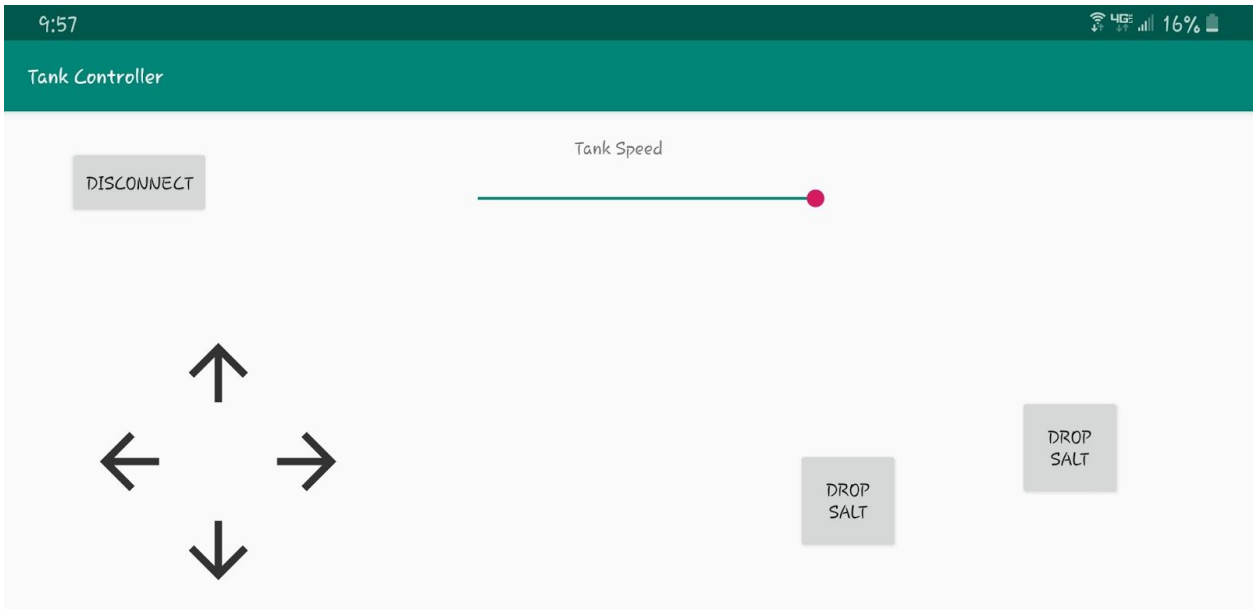
Sketches:

Tank:



Mobile App

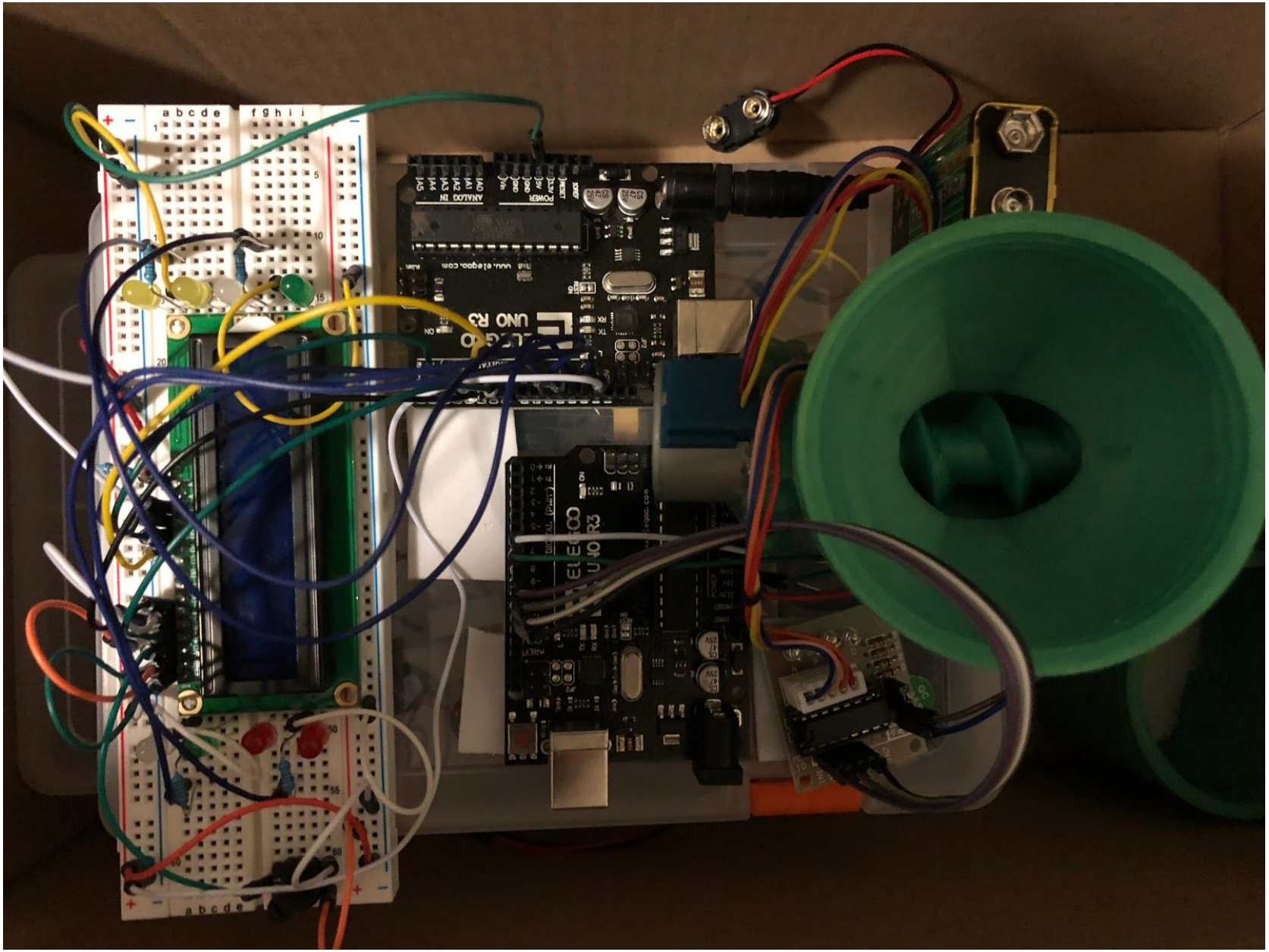

9:56
16%
Tank Controller
Choose Device
Galaxy Buds (5233)
F4:7D:EF:A9:52:33
HC-06
98:D3:61:F9:42:13
AKG N60NC Wireless
34:DF:2A:58:4C:64


9:57
16%
Tank Controller
DISCONNECT
Tank Speed
DROP SALT
DROP SALT

Base:

(Salt Storage) **Arduino 3**
(Motor Controller) (Base)
(Distance Sensor)

High Level Sketch on next page


UNO R3
ELEGOO

```
//Reference from - Written By Nikodem Bartnik - nikodembartnik.pl
//Author - Harsh Devprakash Gupta
//Description: Code for the Base which includes the stepper motor and the ultrasonic sensor. When the tank is empty it
//             arrives near the base and the distance is calculated by the ultrasonic sensor attached to pin 6 and 7.
//             When the distance in inches is less than 2 inches and in cm less than 8 cm then the stepper motor starts and
//             dispenses salt which is added to the tank.
// Libraries for stepper motor and ultrasonic sensor
#define STEPPER_PIN_1 9
#define STEPPER_PIN_2 10
#define STEPPER_PIN_3 11
#define STEPPER_PIN_4 12
const int pingPin = 7; // Trigger Pin of Ultrasonic Sensor
const int echoPin = 6; // Echo Pin of Ultrasonic Sensor

int step_number = 0;
void setup() {
pinMode(STEPPER_PIN_1, OUTPUT);
pinMode(STEPPER_PIN_2, OUTPUT);
pinMode(STEPPER_PIN_3, OUTPUT);
pinMode(STEPPER_PIN_4, OUTPUT);
Serial.begin(9600); // Starting Serial Terminal

}

void loop() {
   long duration, inches, cm;
   pinMode(pingPin, OUTPUT);
   digitalWrite(pingPin, LOW);
   delayMicroseconds(2);
   digitalWrite(pingPin, HIGH);
```

```
    digitalWrite(pingPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(pingPin, LOW);
    pinMode(echoPin, INPUT);
    duration = pulseIn(echoPin, HIGH);
    inches = microsecondsToInches(duration);
    cm = microsecondsToCentimeters(duration);
    Serial.print(inches);
    Serial.print("in, ");
    Serial.print(cm);
    Serial.print("cm");
    Serial.println();
    if ((inches <= 1 && cm < 5) || (inches = 918 && cm >=2342)){
     OneStep(false);
    delay(2);
    }



}
long microsecondsToInches(long microseconds) {
   return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
   return microseconds / 29 / 2;
}


void OneStep(bool dir){
    if(dir){
```

```
58
59  void OneStep(bool dir){
60      if(dir){
61  switch(step_number){
62    case 0:
63    digitalWrite(STEPPER_PIN_1, HIGH);
64    digitalWrite(STEPPER_PIN_2, LOW);
65    digitalWrite(STEPPER_PIN_3, LOW);
66    digitalWrite(STEPPER_PIN_4, LOW);
67    break;
68    case 1:
69    digitalWrite(STEPPER_PIN_1, LOW);
70    digitalWrite(STEPPER_PIN_2, HIGH);
71    digitalWrite(STEPPER_PIN_3, LOW);
72    digitalWrite(STEPPER_PIN_4, LOW);
73    break;
74    case 2:
75    digitalWrite(STEPPER_PIN_1, LOW);
76    digitalWrite(STEPPER_PIN_2, LOW);
77    digitalWrite(STEPPER_PIN_3, HIGH);
78    digitalWrite(STEPPER_PIN_4, LOW);
79    break;
80    case 3:
81    digitalWrite(STEPPER_PIN_1, LOW);
82    digitalWrite(STEPPER_PIN_2, LOW);
83    digitalWrite(STEPPER_PIN_3, LOW);
84    digitalWrite(STEPPER_PIN_4, HIGH);
85    break;
86  }
87    }else{
88      switch(step_number){
```

```
88        switch(step_number){
89      case 0:
90      digitalWrite(STEPPER_PIN_1, LOW);
91      digitalWrite(STEPPER_PIN_2, LOW);
92      digitalWrite(STEPPER_PIN_3, LOW);
93      digitalWrite(STEPPER_PIN_4, HIGH);
94      break;
95      case 1:
96      digitalWrite(STEPPER_PIN_1, LOW);
97      digitalWrite(STEPPER_PIN_2, LOW);
98      digitalWrite(STEPPER_PIN_3, HIGH);
99      digitalWrite(STEPPER_PIN_4, LOW);
100     break;
101     case 2:
102     digitalWrite(STEPPER_PIN_1, LOW);
103     digitalWrite(STEPPER_PIN_2, HIGH);
104     digitalWrite(STEPPER_PIN_3, LOW);
105     digitalWrite(STEPPER_PIN_4, LOW);
106     break;
107     case 3:
108     digitalWrite(STEPPER_PIN_1, HIGH);
109     digitalWrite(STEPPER_PIN_2, LOW);
110     digitalWrite(STEPPER_PIN_3, LOW);
111     digitalWrite(STEPPER_PIN_4, LOW);
112
113
114 }
115   }
116 step_number++;
117   if(step_number > 3){
118     step_number = 0;
```

```
    digitalWrite(STEPPER_PIN_3, HIGH);
    digitalWrite(STEPPER_PIN_4, LOW);
    break;
    case 2:
    digitalWrite(STEPPER_PIN_1, LOW);
    digitalWrite(STEPPER_PIN_2, HIGH);
    digitalWrite(STEPPER_PIN_3, LOW);
    digitalWrite(STEPPER_PIN_4, LOW);
    break;
    case 3:
    digitalWrite(STEPPER_PIN_1, HIGH);
    digitalWrite(STEPPER_PIN_2, LOW);
    digitalWrite(STEPPER_PIN_3, LOW);
    digitalWrite(STEPPER_PIN_4, LOW);


}
  }
step_number++;
  if(step_number > 3){
    step_number = 0;
  }
}
```

```
//Reference From - https://www.arduino.cc/en/Tutorial/HelloWorld
//Author - Brian Goldenberg
//Description: Code used to display base station through LCD display
//             Lights up multiple leds around display to emphasize
//Libraties for Liquid crystal display



#include <LiquidCrystal.h>

// Setup code used for initliazing
  const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
  LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
    const int ledPin1 = 13;
    const int ledPin2 = 12;
  const int ledPin3 = 11;
  const int ledPin4 = 10;


void setup() {
  // initialize digital pins as outputs



  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(ledPin1,OUTPUT);
  pinMode(ledPin2,OUTPUT);
  pinMode(ledPin3,OUTPUT);
  pinMode(ledPin4,OUTPUT);

```

```
 //Display to LCD
  lcd.begin(16, 2);
  lcd.print("BASE STATION!");
  delay(1000);
}

// the loop function runs over and over again forever
void loop() {
 //Loop function used for switching LEDS to on and oof


 //Switch to on
  digitalWrite(ledPin1,HIGH);
  digitalWrite(ledPin2,HIGH);
  digitalWrite(ledPin3,HIGH);
  digitalWrite(ledPin4,HIGH);



  delay(1000);                        // wait for a second
      // turn the LED off by making the voltage LOW
   digitalWrite(ledPin1,LOW);
  digitalWrite(ledPin2,LOW);
  digitalWrite(ledPin3,LOW);
  digitalWrite(ledPin4,LOW);



  delay(1000);                        // wait for a second
}
```