PiRover

By Christopher Albarillo, Lawrence Puig, and Patrick Ng

April46 23, 2018

Project Build Website: https://chris0707.github.io/PiRover/

Declaration of Joint Authorship

We, the PiNivea group, here by declare that the project report entitled “PiRover” confirm that this work submitted for assessment is our own and is expressed in our own words. Any sources and material used are all cited and documented. A list of references that were used for this project is provided and can be found under Technical References section.

PiNivea: Christopher Albarillo, Lawrence Puig, and Patrick Ng.

Date: February 23, 2017

Proposal

Executive Summary

As students in the Computer Engineering Technology program, we will be integrating the knowledge and skills we have learned from our program. This proposal requests the approval to build the hardware portion, which is the PiRover, that will connect to a mobile device application using Bluetooth connection. The mobile device functionality will allow the user to register and login their information, gain access wirelessly to control the hardware(PiRover), voice commands to control the PiRover's features (lights and code execution).

Background

We believe that the most difficult part of this project is building the script that will handle the voice command functionality that will control the PiRover's pre-implemented features such as; lights and other code executions.

This prototype project that our team has constructed has been inspired by other rovers such as NASA's own rover. By utilizing all of the abilities acquired in our previous Hardware course we have gained further knowledge and become accustomed to the further advancements we are applying into the prototype. The PiRover was designed to be controlled via Bluetooth with an android application from any android device. With this feature working, we plan to implement other features such as; lights, voice activations and maybe further perfect the controller from the android application.

Methodology

This proposal is assigned in the first week of class and is due at the beginning of class in the second week of the winter semester. My coursework will focus on the first two of the 3 phases of this project:

Phase 1 Hardware build.

Phase 2 System integration.

Phase 3 Demonstration to future employers.

Phase 1 Hardware build

The hardware build has been completed in the fall term and will finish it with an enclosed case for protection and stability this winter term. It will fit within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that will be used is 16Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will be 20 Watts.

Phase 2 System integration

The system integration will be completed in the winter term.

Phase 3 Demonstration to future employers

This project will showcase the knowledge and skills that I have learned to potential employers.

The brief description below provides rough effort and non-labour estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines.

Labour Estimate	**Hrs**	**Notes**
Phase 1		
Writing Proposal	9	Tech Identification Quiz
Creating Project Schedule	9	Proposal Due
Creating Budget (Mock group setup for CENG355/Phase 2)	9	Project Schedule Due
Acquiring Components / Begin to film unboxing	9	Project Budget Due

PCB Fabrication	9	Component received for the project
Mechanincal Assembly / Status Meeting	9	Showing acquisition for the project
Finish off the PCB	9	Mechanical Assembly Due
Writing 30 second Video Script and creating Placard Design	9	PCB Due
Creating 30 second Video with the script created and Demonstration the hardware	9	Video Script and Placard Due
Writing Progress Report	9	30 second Video Due
Creating the Hardware Presentation	9	Progress Report Due
1st round of presentation and Writing build report	9	Presentation Due
2nd round of presentation	9	Build Report Due

Phase 1 Total	**117**	
Phase 2		
Recreate the Proposal	9	Form the group for the project
Meeting with collaborators (SRS)	9	Proposal Due
Status Meeting	9	SRS Due
Meeting with collaborators(Abstract, Introduction and Declaration of Authorship)	9	Family Day Holiday – No Class
Meeting with collaborators (Email Progres Report by Student A)	9	Abstract, Introduction and Declaratiob of Authorship Due
Meeting with collaborators (Merged Build Instruction ported to Technical Report and App, Web and	9	Group Status Progress Report Due

Database Independent Demonstration)		
Meeting with collaborators(Email Progress Report by Student B)	9	Merged Build Instruction ported to Technical Report and App, Web and Database Independent Demonstration Due
Meeting with collaborators(OACETT basic requirement report checklist)	9	Integration Progress Report Due
Meeting with collaborators(Email Progress Report by Student C)	9	OACETT basic requirement report checklist Due
Prepare for Demonstration	9	Troubleshooting Progress Report Due
Prepare for Presentation	9	Demonstration Due
Writing Technical Report	9	Presentation Due
Extra Day	N/A	Technical Report Due

Phase 2 Total	**108**	
Phase 3		
Interviews	TBD	
Phase 3 Total	**TBD**	
Material Estimate	**Cost with tax**	**Quantity**
Raspberry PI Starter Kit	$112.99	1
Electronic Learning Kit for RaspPi	$19.05	1
100000mAh Portable Power Bank	$23.72	1
Micro Servo Motor FS90R	$21.00	2
Ultrasonic Sensor HC-SR04	$5.00	1
Laserut Chasis from Humber	N/A	1
3D Printed Wheels from Humber	N/A	1

Total	$181.67	

Concluding Remarks

This proposal presents a plan for providing an IoT solution for search and rescue situations or even for historical research. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating our ability to learn how to support projects such as the initiative described by T. Kubota, Y. Kuroda, Y. Kunii and T. Yoshimitsu, "Path planning for newly developed microrover," Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164), 2001, pp. 3710-3715 vol.4.
doi: 10.1109/ROBOT.2001.933195
keywords: {computerised navigation;data structures;microrobots;mobile robots;path planning;planetary rovers;probability;data structure;elevation map;mobile robots;navigation;path planning;planetary exploration;planetary microrover;probability;Data structures;Instruments;Mars;Mobile robots;Moon;NASA;Navigation;Path planning;Rain;Space exploration},
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=933195&isnumber=20185

We request approval of this project.

Abstract

Technology is growing fast and is considered to be a powerful tool to help us in our daily lives. One of these technologies that lead us to our project is the rover, that can be controlled wirelessly and also can fit in small and tight spaces which no average person can fit into. In addition, it gives the researchers comfortability to not worry about the dangers of a certain place that they will explore. Having a rover can benefit a lot of situations such as: outdoor research, emergency situations, and for learning purposes. However, with such technology that has a lot to offer, also comes with an incredible amount of price. This is where our project will come to play. Our project, PiRover, is a portable rover that is powered by an open source computer called Raspberry Pi. In this project, we focused on the portability, low cost, and functionality of the PiRover. This gives the people opportunity to get themselves their own rover for a lower cost to explore and learn using the PiRover at the same time.

Table of Contents

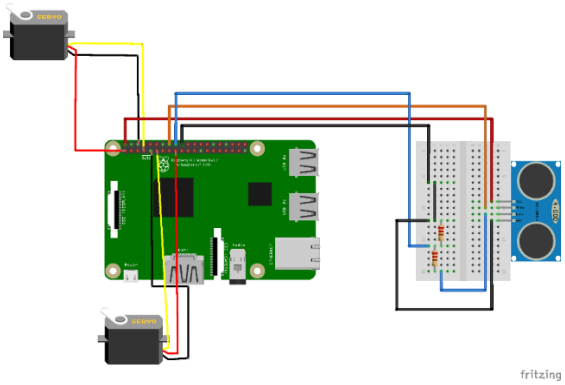
1. Illustration List

1.1 Definitions, Acronyms, and Abbreviations

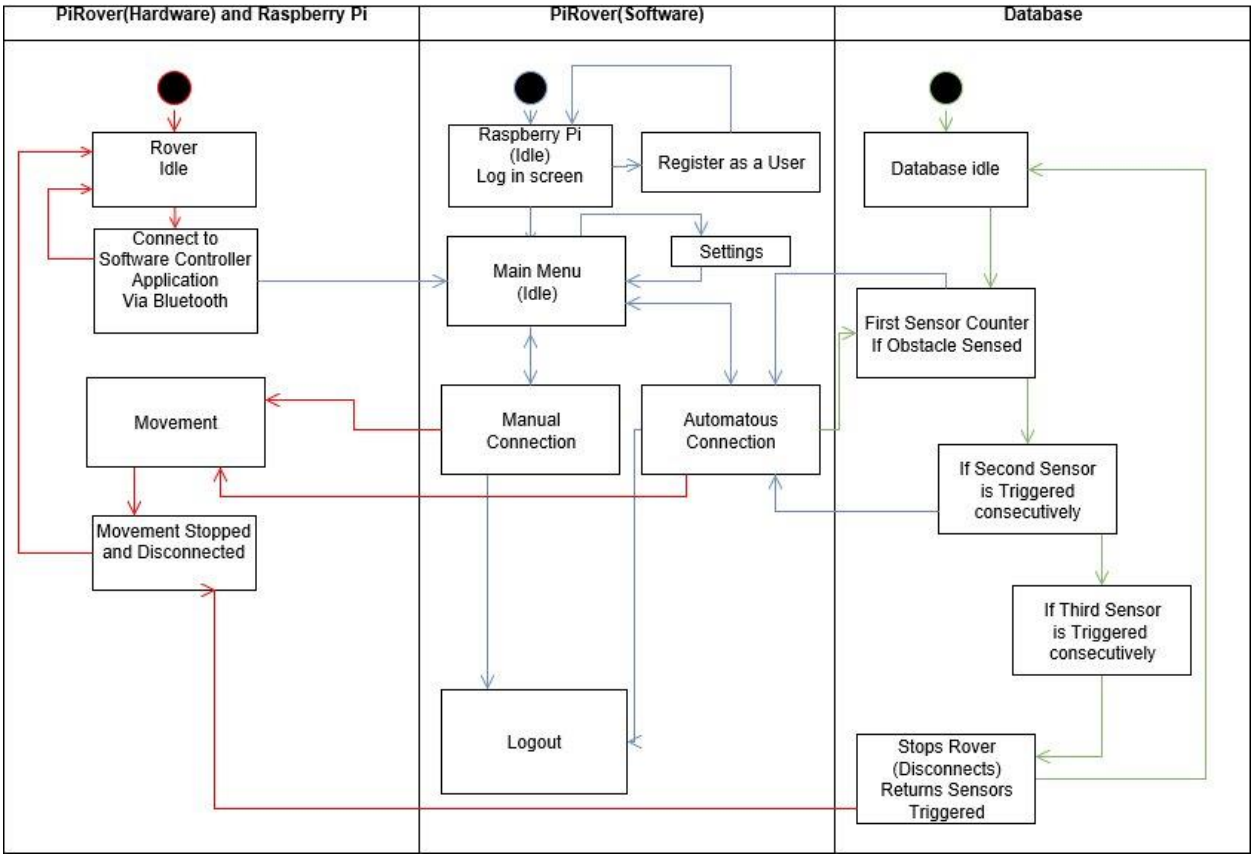
TERM	DEFINITION
PIROVER	Title of the project and name of both the prototype and android application
PCB	Printed Circuit Board – Used to compacts the connections into a flat board
BLUETOOTH	Used to Connect wirelessly from device to device to exchange data
DATABASE	Holds information in a server, such as; security information
GUI	Graphical User Interface – an interface that allows users to access functions in a user friendly manner.
DATABASE	A structured set of data held in a computer or server
ANDROID STUDIO	The IDE that can be downloaded from Google, used to create mobile application
IDE	Integrated Development Environment

1.2 System Diagram

Overview of the system diagram:



1.3 UML Diagram



2. Introduction

2.1 Purpose of Documentation

The purpose of this documentation is to give a detailed outlook of the prototype, “PiRover”, and the requirements, hardware, software and other features associated with the prototype. This documentation is primarily intended as a reference and for the development of the PiRover.

2.2 Scope

The PiRover is a micro rover prototype used to explore area such as small/tight spaces which the average person could not fit into or areas that are dangerous. For example, outer space, scientists can use this to explore areas that they deem not safe for humans. The hardware will contain two FS90R micro servo motor to rotate the wheels, HC-SR04 Ultrasonic Sensor to send out a pressure wave to detect foreign objects in its path. PiRover also has an Android application where the user can control the rover wireless via Bluetooth. Due to the application only being Android, this restricts any other users that don’t have Android device that is compatible to 5.0 (API 21)

Once the hardware is complete and the Android application is installed on your device, you will need to register in order to use the application with certain preferences that you would like to be saved. Afterward, when you have login with the newly created account, it will prompt what type of control that the user wants and they can start using it. The hardware will send data to the database if it is connected to the internet. An Internet connection is not required in order to use this application but is recommended.

The database information stored from the hardware will be visible to the user on their Android device. On the mobile application, the user has to be logged in to access this information as well as having internet connection to allow the application to pull the information from the database.

3 Software Requirement Specifications (SRS)

3.1 Purpose

The purpose of our software is to provide full control of the PiRover(Hardware). Using the mobile application, it enables the user to control the PiRover manually and automatically and have his/her own account to simply store the PiRover's collected data in the database.

3.2 Overall Description

3.2.1 System Interface

The system interface for our project consists of the following: PiRover hardware, mobile application, and a database server. The user will have full control of the PiRover using the mobile application. When the user chooses the option to control the PiRover automatically, it will run autonomously while recording data and having an option to review and store the information in the database.

3.3 Hardware

Our hardware is called PiRover. Once the PiRover has booted up, it will automatically open a Bluetooth server connection. It should be able to receive the connection from the mobile device that the hardware is currently paired with. Once the connected is established, the user is can now control the PiRover manually and automatically. Patrick Ng will be leading this section.

3.3.1 Components

PiRover Components:

- Raspberry Pi 3
- UltraSonic Sensor HC-SR04
- 100000mAh Portable Power Bank
- Micro Servo Motor FS90R (x2)
- Laser cut Chasis from Humber College
- 3D Printed wheels from Humber College
- PCB Printed from Humber College
- USB Microphone

3.3.2 Chasis Specifications

Chasis Size:

Width – 8.99cm

Height – 5.99cm

3.3.3 Motor and PCB(Optional) Specifications

Motor Specifications:

Size:

23.2mm x 12.5mm x 22mm

General Specifications (at 6V):

Free-run current: 120 mA

Stall current: 800 mA

Speed: 130 rpm

Stall Torque: 21 oz-in

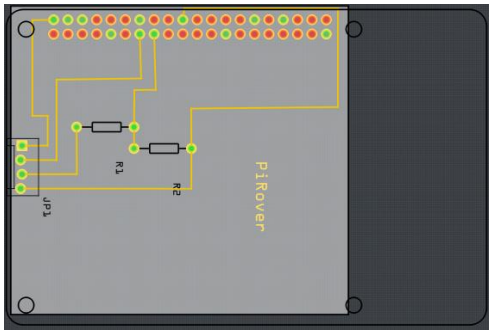
General Specifications (at 4.8V):

Speed: 100 rpm

Stall torque 18 oz-in

Lead Length: 10 in

PCB Specifications:



PCB Size:

Width – 55mm

Height – 60mm

3.4 Mobile Application

Our mobile application is called PiRover controller. The mobile application is only available for android users only. Once the app is opened, it will give the user options to – login, register, and proceed in offline mode. After the user has logged in, there will be two control options provided for the user: manual and automatic. If the user selected manual, he/she will have the ability to control the PiRover using the manual controls provided. And if the user selected automatic, he/she will only have one option to control the PiRover, and that is by selecting or clicking the button “start automap”. Lawrence Puig will be leading this section.

3.4.1 Database

The database server that is being used in this project is the Hostinger's database. Using the database, it enables us to store user's information, as well as the data that will be gathered by the PiRover. This will give the user comfortability to review their gathered data as long as the user has access to the internet and currently logged on to the mobile application. Christopher Albarillo will be leading this section.

3.4.2 Connectivity

The mobile application allows the user to graphically control the PiRover via Bluetooth integration to the Raspberry Pi 3. The Android device and Raspberry Pi 3 should be paired in order to control the PiRover. For further details refer to the Mobile application's "Help" section.

3.4.3 Manual connectivity code

ManualActivity.java

```
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.support.constraint.ConstraintLayout;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.Toast;

import com.example.chris.piroverapp.R;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Set;
import java.util.UUID;


import static java.lang.System.exit;

public class ManualActivity extends AppCompatActivity {
```

```
    private android.os.Handler handler = new android.os.Handler();

    ImageButton forwardButton, reverseButton, leftButton, rightButton;
    Button dc;


    BluetoothAdapter blueAdapter;
    BluetoothSocket blueSocket;
    BluetoothDevice blueDevice;

    OutputStream outputStream;
    InputStream inputStream;

    String a;
    String message="";
    String colour;
    boolean connected;
    boolean runThreadRunning = false;
    boolean runThreadStop = false;
    ConstraintLayout cons;




    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_manual);


        dc = (Button) findViewById(R.id.buttonDc);
        final Button connect = (Button) findViewById(R.id.buttonBlue);


        connect.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                //a = "raspberrypi";
                a = getString(R.string.raspberrypiID);
                try {
                    if (isBluetoothAvailable()) {
                        try {
                            findBT();
                            Toast.makeText(getBaseContext(),
R.string.trying, Toast.LENGTH_SHORT).show();
                        } catch (IOException e) {
                            Toast.makeText(getBaseContext(),
R.string.notestablished, Toast.LENGTH_SHORT).show();
                            e.printStackTrace();
                        }
                        Toast.makeText(getBaseContext(),
R.string.connected, Toast.LENGTH_SHORT).show();
                        connected = true;
                    } else {
                        Toast.makeText(getBaseContext(),
R.string.bluetoothelse, Toast.LENGTH_SHORT).show();
```

```
                    }
                }catch (NullPointerException e){
                   e.printStackTrace();
                    Toast.makeText(getBaseContext(),
R.string.noconnection, Toast.LENGTH_SHORT).show();

                }


            }
        });


        dc.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (connected) {
                    Toast.makeText(getBaseContext(), R.string.closing,
Toast.LENGTH_SHORT).show();
                    String stop = getString(R.string.stopButton);
                    sendMsg(stop);
                    exit(0);

                    connected = false;
                } else {
                    Toast.makeText(getBaseContext(),
R.string.noconnection, Toast.LENGTH_SHORT).show();

                }
            }
        });


        forwardButton = (ImageButton) findViewById(R.id.forwardButton);
        forwardButton.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent motionEvent) {
                if(connected){
                switch (motionEvent.getAction()) {
                    case MotionEvent.ACTION_DOWN: {
                        String w = getString(R.string.forward);

                        handleRunDown(w);
                        return true;
                    }
                    case MotionEvent.ACTION_UP: {
                        handleRunUp();
                        return true;
                    }
                }}

                return false;
            }

        });
```

```
/*forwardButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if(connected)
            sendMsg("w");

        int duration = Toast.LENGTH_SHORT;
        Context context = getApplicationContext();
        int text = R.string.m_forward;
        Toast toast = Toast.makeText(context,text,duration);
        //toast.show();
    }
}); */

reverseButton = (ImageButton) findViewById(R.id.reverseButton);
reverseButton.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if(connected){
            switch (motionEvent.getAction()) {
                case MotionEvent.ACTION_DOWN: {

                    String s = getString(R.string.reverse);

                    handleRunDown(s);
                    return true;
                }
                case MotionEvent.ACTION_UP: {
                    handleRunUp();
                    return true;
                }
            }}

        return false;
    }

});
/* reverseButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (connected)
            sendMsg("s");
        int duration = Toast.LENGTH_SHORT;
        Context context = getApplicationContext();
        int text = R.string.m_reverse;
        Toast toast = Toast.makeText(context, text, duration);
        //toast.show();
    }
});*/

leftButton = (ImageButton) findViewById(R.id.leftButton);
leftButton.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if(connected){
            switch (motionEvent.getAction()) {
                case MotionEvent.ACTION_DOWN: {
```

```
                        String a = getString(R.string.left);

                        handleRunDown(a);
                        return true;
                    }
                    case MotionEvent.ACTION_UP: {
                        handleRunUp();
                        return true;
                    }
                }}

            return false;
        }

    });
   /* leftButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (connected)
                sendMsg("a");
            int duration = Toast.LENGTH_SHORT;
            Context context = getApplicationContext();
            int text = R.string.m_left;
            Toast toast = Toast.makeText(context, text, duration);
            //toast.show();
        }
    });*/

    rightButton = (ImageButton) findViewById(R.id.rightButton);
    rightButton.setOnTouchListener(new View.OnTouchListener() {
        @Override
        public boolean onTouch(View view, MotionEvent motionEvent) {
            if(connected){
                switch (motionEvent.getAction()) {
                    case MotionEvent.ACTION_DOWN: {

                        String d = getString(R.string.right);

                        handleRunDown(d);
                        return true;
                    }
                    case MotionEvent.ACTION_UP: {
                        handleRunUp();
                        return true;
                    }
                }}

            return false;
        }

    });
  /* rightButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (connected)
                sendMsg("d");
```

```
                int duration = Toast.LENGTH_SHORT;
                Context context = getApplicationContext();
                int text = R.string.m_right;
                Toast toast = Toast.makeText(context, text, duration);
                //toast.show();
            }
        });*/

    }



    public static boolean isBluetoothAvailable() {
        final BluetoothAdapter bluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();
        return (bluetoothAdapter != null &&
bluetoothAdapter.isEnabled());
    }

    void findBT() throws IOException {

        blueAdapter = BluetoothAdapter.getDefaultAdapter();

        if (blueAdapter == null) {
            exit(0);

        }

        if (!blueAdapter.isEnabled()) {
            Intent enableBluetooth = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBluetooth, 0);

        }

        Set<BluetoothDevice> pairedDevices =
blueAdapter.getBondedDevices();
        if (pairedDevices.size() > 0) {

            for (BluetoothDevice device : pairedDevices) {
                if (device.getName().equals(a)) {

                    blueDevice = device;
                    break;
                }

            }

        }
        String serial = getString(R.string.register_serialButton1);
        //UUID uuid = UUID.fromString("00001101-0000-1000-8000-
00805f9b34fb");
        UUID uuid = UUID.fromString(serial);

        blueSocket = blueDevice.createRfcommSocketToServiceRecord(uuid);
        try {
            blueSocket.connect();
```

```
        } catch (IOException e) {
            e.printStackTrace();
            exit(0);
        } finally {
            outputStream = blueSocket.getOutputStream();
            inputStream = blueSocket.getInputStream();
        }

    }

    void sendMsg(String msg1) {
        try {
            outputStream.write(msg1.getBytes());
        } catch (IOException e) {
            e.printStackTrace();
            //exit(0);
        }
    }

    private void handleRunDown(String mes) {
        if (!runThreadRunning)
            startRunThread(mes);
    }

    private void startRunThread(final String mes) {
        Thread r = new Thread() {

            @Override
            public void run() {
                try {
                    runThreadRunning = true;
                    while (!runThreadStop) {
                        handler.post(new Runnable() {
                            @Override
                            public void run() {
                                sendMsg(mes);

                                //keepSending(message);
                            }
                        });
                        try {
                            Thread.sleep(50);

                        } catch (InterruptedException e) {
                            throw new RuntimeException("Could not
wait...", e);
                        }

                    }
                } finally {
                    runThreadRunning = false;
                    runThreadStop = false;
                }
            }
        };
```

```
        r.start();
    }

    private void handleRunUp(){
        runThreadStop = true;
    }

   /* private void keepSending(String message){
        sendMsg(message);
    }*/

}
```

3.4.4 Automonous connectivity code

AutoActivity.java

```
import android.annotation.SuppressLint;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.example.chris.piroverapp.R;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.text.DateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.UUID;
```

```
import static java.lang.System.exit;

public class
AutoActivity extends AppCompatActivity {
    private android.os.Handler handler = new android.os.Handler();

    BluetoothAdapter blueAdapter;
    BluetoothSocket blueSocket;
    BluetoothDevice blueDevice;

    OutputStream outputStream;
    InputStream inputStream;


    Button toggleButton;
    Button connect, dc, startB, saveData, stopB;
    Boolean check = Boolean.FALSE;
    TextView viewData;

    String a;
    String message="";
    boolean connected;
    boolean runThreadRunning = false;
    boolean runThreadStop = false;

    private static final String URL =
"http://thebarebarzz.xyz/UpdateCheck.php";
    private RequestQueue requestQueue;
    private StringRequest request;
    String userName;
    String currentDateTimeString;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_auto);



        connect = (Button)findViewById(R.id.buttonBlue);
        dc = (Button)findViewById(R.id.buttonDc);
        startB =(Button)findViewById(R.id.startButton);
       // stopB = (Button)findViewById(R.id.stopButton);
        saveData = (Button) findViewById(R.id.saveDataButton);
        viewData = findViewById(R.id.viewText);
        connect.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //a = "raspberrypi";
                a = getString(R.string.raspberrypiID);
                try {
                    if (isBluetoothAvailable()) {
                        try {
                            findBT();
                            Toast.makeText(getBaseContext(),
R.string.trying, Toast.LENGTH_SHORT).show();
                        } catch (IOException e) {
```

```
                            Toast.makeText(getBaseContext(),
R.string.notestablished, Toast.LENGTH_SHORT).show();
                            e.printStackTrace();
                        }
                        Toast.makeText(getBaseContext(),
R.string.connected, Toast.LENGTH_SHORT).show();
                        connected = true;
                    } else {
                        Toast.makeText(getBaseContext(),
R.string.bluetoothelse, Toast.LENGTH_SHORT).show();
                    }
                }catch (NullPointerException e){
                    e.printStackTrace();
                    Toast.makeText(getBaseContext(),
R.string.noconnection, Toast.LENGTH_SHORT).show();

                }
            }
        });

        dc.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (connected) {
                    Toast.makeText(getBaseContext(), R.string.closing,
Toast.LENGTH_SHORT).show();
                    //sendMsg("stop");
                    String stop = getString(R.string.stopButton);
                    sendMsg(stop);
                    exit(0);

                    connected = false;
                } else {
                    Toast.makeText(getBaseContext(),
R.string.noconnection, Toast.LENGTH_SHORT).show();

                }
            }
        });

        startB.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(connected){
                    String start = getString(R.string.startAuto);
                    sendMsg(start);
                    getMsg();


                } else{
                    Toast.makeText(AutoActivity.this,
getResources().getString(R.string.toast_autoMap),
Toast.LENGTH_SHORT).show();
                }
            }
        });
```

```
        requestQueue = Volley.newRequestQueue(this);
        saveData.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                final String sendData = (String) viewData.getText() +
currentDateTimeString;
                final String userNameString;
              /*  Intent intent = getIntent();
                Bundle bundle = intent.getExtras();

                if(bundle != null){
                    userName = bundle.getString("username");
                }*/
                currentDateTimeString = "
"+DateFormat.getDateTimeInstance().format(new Date());
                userName = getIntent().getStringExtra("username2");

                //test for update database
                //viewData.setText(userName);
                request = new StringRequest(Request.Method.POST, URL,
new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        JSONObject jsonObject = null;
                        try {
                            jsonObject = new JSONObject(response);
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }

                        try {
                            assert jsonObject != null;
                            if
(jsonObject.names().get(0).equals("success")) {
                                Toast.makeText(getApplicationContext(),
"Update Success! " + jsonObject.getString("success"),
Toast.LENGTH_SHORT).show();


                            }else{
                                Toast.makeText(getApplicationContext(),
"Update Error! " + jsonObject.getString("error"),
Toast.LENGTH_SHORT).show();

                            }
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }

                    }
                }, new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {

                    }
```

```
                }) {
                    @Override
                    protected Map<String, String> getParams() throws
AuthFailureError{
                        HashMap<String,  String> hashMap = new
HashMap<String, String>();
                        hashMap.put("username", userName);
                        hashMap.put("content", sendData);

                        return hashMap;
                    }
                };
                requestQueue.add(request);


            }
        });



    /*    stopB.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(connected){
                    String stop = getString(R.string.stopAuto);
                    sendMsg(stop);

                }else{
                    Toast.makeText(AutoActivity.this,
getResources().getString(R.string.toast_autoStop),
Toast.LENGTH_SHORT).show();
                }
            }
        });*/


    }


    public static boolean isBluetoothAvailable() {
        final BluetoothAdapter bluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();
        return (bluetoothAdapter != null &&
bluetoothAdapter.isEnabled());
    }

    void findBT() throws IOException {

        blueAdapter = BluetoothAdapter.getDefaultAdapter();

        if (blueAdapter == null) {
            exit(0);

        }

        if (!blueAdapter.isEnabled()) {
```

```
            Intent enableBluetooth = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBluetooth, 0);

        }

        Set<BluetoothDevice> pairedDevices =
blueAdapter.getBondedDevices();
        if (pairedDevices.size() > 0) {

            for (BluetoothDevice device : pairedDevices) {
                if (device.getName().equals(a)) {

                    blueDevice = device;
                    break;
                }

            }

        }

        //UUID uuid = UUID.fromString("00001101-0000-1000-8000-
00805f9b34fb");
        String serial = getString(R.string.register_serialButton1);
        UUID uuid = UUID.fromString(serial);

        blueSocket = blueDevice.createRfcommSocketToServiceRecord(uuid);
        try {
            blueSocket.connect();

        } catch (IOException e) {
            e.printStackTrace();
            exit(0);
        } finally {
            outputStream = blueSocket.getOutputStream();
            inputStream = blueSocket.getInputStream();
        }
        //test


    }


    void sendMsg(String msg1) {
        try {
            outputStream.write(msg1.getBytes());
        } catch (IOException e) {
            e.printStackTrace();
            //exit(0);
        }
    }

    void getMsg(){
        byte[] buffer = new byte[1024];
        int bytes;
        final String obstacleText = getString(R.string.auto_dataTitle);
```

```
        try{
            bytes = inputStream.read(buffer);
            final String incomingMessage = new String(buffer, 0, bytes);
            Log.d("AutoActivity", "InputStream: " + incomingMessage);

            viewData.setText(incomingMessage);
        }catch (IOException e){
            e.printStackTrace();

        }

    }

  /*  private void handleRunDown(String mes) {
        if (!runThreadRunning)
            startRunThread(mes);
    }*/

    private void startRunThread(final String mes) {
        Thread r = new Thread() {

            @Override
            public void run() {
                try {
                    runThreadRunning = true;
                    while (!runThreadStop) {
                        handler.post(new Runnable() {
                            @Override
                            public void run() {
                                sendMsg(mes);

                                //keepSending(message);
                            }
                        });
                        try {
                            Thread.sleep(50);

                        } catch (InterruptedException e) {
                            throw new RuntimeException("Could not
wait...", e);
                        }

                    }
                } finally {
                    runThreadRunning = false;
                    runThreadStop = false;
                }
            }
        };

        r.start();
    }



}
```

3.5 Hardware Code

3.5.1 Ultrasonic sensor python code

sensorA.py

```
import RPi.GPIO as gpio
import time

def distance(measure='cm'):
    gpio.setmode(gpio.BOARD)
    gpio.setup(16,gpio.OUT)
    gpio.setup(18,gpio.IN)

#while True:
    gpio.output(16, True)
    time.sleep(0.00001)
    gpio.output(16, False)

    startTime = time.time()
    stopTime = time.time()

    while gpio.input(18) == 0:
        startTime = time.time()

    while gpio.input(18) == 1:
        stopTime = time.time()

    distance = (stopTime - startTime)*17000


    return distance

    print("distance: %.1f cm" % distance)
    #time.sleep(1)
```

3.5.2 Autonomous mode without the mobile application python code

servoAuto.py

```
import RPi.GPIO as gpio
import time
import sys
import Tkinter as tk
from sensorA import distance
import random
```

```
gpio.setmode(gpio.BOARD)
gpio.setup(7, gpio.OUT)
gpio.setup(11, gpio.OUT)
pwm=gpio.PWM(7,50)
pwm2=gpio.PWM(11,50)
pwm2.start(0)
pwm.start(0)


def forward(tf):
    DC=1./18.*(1)+2
    DC2=1./18.*(180)+2
    pwm.ChangeDutyCycle(DC2)
    pwm2.ChangeDutyCycle(DC)
    time.sleep(tf)
    pwm.start(0)
    pwm2.start(0)

def reverse(tf):
    DC=1./18.*(1)+2
    pwm.ChangeDutyCycle(DC)
    DC2=1./18.*(180)+2
    pwm2.ChangeDutyCycle(DC2)
    time.sleep(tf)
    pwm.start(0)
    pwm2.start(0)

def turnLeft(tf):
    #for i in range(0,180):
        DC=1./18.*(1)+2
        pwm.ChangeDutyCycle(DC)
        pwm2.ChangeDutyCycle(DC)
        time.sleep(tf)
        pwm2.start(0)
        pwm.start(0)
        #gpio.cleanup()

def turnRight(tf):
        DC=1/18.*180+2
        pwm.ChangeDutyCycle(DC)
        pwm2.ChangeDutyCycle(DC)
        time.sleep(tf)
        pwm2.start(0)
        pwm.start(0)
        #gpio.cleanup()


'''def key_input(event):

    print 'Key: ', event.char
    key_press = event.char
    s = 0.030

    if key_press.lower()=='w':
        forward(s)
    elif key_press.lower()=='s':
        reverse(s)
```

```
    elif key_press.lower()=='a':
        turnLeft(s)
    elif key_press.lower()=='d':
        turnRight(s)


command = tk.Tk()
command.bind('<KeyPress>', key_input)
command.mainloop()
'''
def check_front():
    dist = distance()

    if dist < 15:
        print('Too close,', dist)

        reverse(0.4)
        turnLeft(1)
        dist = distance()

        if dist < 15:
            print('Too close. im done.', dist)
            sys.exit()

def autonomy():
    tf = 0.030
    x = random.randrange(0,4)

    if x==0:
        for y in range(30):
            check_front()
            forward(tf)
    elif x == 1:
        for y in range(30):
            check_front()
            forward(tf)
    elif x == 2:
        for y in range(30):
            check_front()
            forward(tf)

for z in range(10):
    autonomy()
```

3.5.3 Autonomous mode with the mobile application python code

```
auto1.py
import RPi.GPIO as gpio
import time
import sys
```

```
import Tkinter as tk
from bluetooth import*
import os
import random



gpio.setmode(gpio.BOARD)
gpio.setup(7, gpio.OUT)
gpio.setup(11, gpio.OUT)
pwm = gpio.PWM(7,50)
pwm2 = gpio.PWM(11,50)
pwm.start(0)
pwm2.start(0)

def forward(tf):
    DC=1./18.*(1)+2
    DC2=1./18.*(180)+2
    pwm.ChangeDutyCycle(DC2)
    pwm2.ChangeDutyCycle(DC)
    time.sleep(tf)
    pwm.start(0)
    pwm2.start(0)

def reverse(tf):
    DC=1./18.*(1)+2
    DC2=1./18.*(180)+2
    pwm.ChangeDutyCycle(DC)
    pwm2.ChangeDutyCycle(DC2)
    time.sleep(tf)
    pwm.start(0)
    pwm2.start(0)

def turnLeft(tf):

    DC=1./18.*(1)+2
    pwm.ChangeDutyCycle(DC)
    pwm2.ChangeDutyCycle(DC)
    time.sleep(tf)
    pwm.start(0)
    pwm2.start(0)

def turnRight(tf):

    DC=1./18.*(180)+2
    pwm.ChangeDutyCycle(DC)
    pwm2.ChangeDutyCycle(DC)
    time.sleep(tf)
    pwm.start(0)
    pwm2.start(0)

def stop():
    time.sleep(1)
    gpio.cleanup()

def check_front():
    dist = distance()
```

```
    if dist<15:
        print('Too close,', dist)

        reverse(1)
        turnLeft(1)

        if dist < 15:
            print("Alright... that's too close. Im done", dist)
            sys.exit()

def autonomy():
    tf = 0.030
    x = random.randrange(0,4)

    if x==0:
        for y in range(30):
            check_front()
            froward(tf)

    elif x==1:
        for y in range(30):
            check_front()
            forward(tf)
    elif x==2:
        for y in range(30):
            check_front()
            forward(tf)



server_sock = BluetoothSocket( RFCOMM )
server_sock.bind(("",PORT_ANY))
server_sock.listen(1)

port = server_sock.getsockname()[1]

uuid = ("00001101-0000-1000-8000-00805f9b34fb");

advertise_service(server_sock, "SampleServer",
        service_id = uuid,
        service_classes = [ uuid, SERIAL_PORT_CLASS ],
        profiles = [ SERIAL_PORT_PROFILE ],
     #   protocols = [ OBEX_UUID ]
        )
print("Waiting for connection on RFCOMM channel %d" % port)

client_sock, client_info = server_sock.accept()
print("Accepted connection from ", client_info)

try:
    while True:
        data = client_sock.recv(1024)
        sleep = 0.030

        if data =='w':
```

```
            print("moving forward")
            forward(sleep)

        elif data == 's':
            print("moving backwards")
            reverse(sleep)

        elif data == 'a':
            print("turning left")
            turnLeft(sleep)

        elif data == 'd':
            print("turning right")
            turnRight(sleep)

        elif data == 'automap':
            print("auto mapping")
            for z in range(10):
                autonomy()


        elif data == 'stop':
            print("stopping")
            stop()


except IOError:
    pass

print("Disconnected")

client_sock.close()
server_sock.close()
print("Done")
```

3.6 External Interface Requirements

In order to run the android application, an Android device that is above API 21 (Android 5.0) is required. Any device under this API will not be able to run the application. For the voice activation control, it will require an external microphone as the PI doesn't have an integrated microphone. This can be a USB webcam with a microphone, USB microphone or 3.5mm microphone. To use the 3.5mm microphone, a USB sound card is needed as the PI 3.5mm jack is an output and not an input. The USB ones are plug and play.

3.7 System Features

The PiRover has multiple features and still going through multiple updates for further implementations. Up to date features include;

- Full Manual Control of the PiRover
- Automatic Sensor Mode

System Features in the process of being implemented;

- Voice Activation Control System *Patrick*
- Lights (May be Voice controlled) *Chris*
- Movement (Possibly voice controlled) *Lawrence*

4 Progress Report

Week 6: Christopher Albarillo (Independent Progress)

Our progress for this week, is as follows:

Christopher - Currently working on our database. I have already created the fields which consists of the Username, Password, Name, and Content. Things that needs to be worked on or improved; password verification, username already exists verification, and hashing of the passwords that are stored.

Lawrence - Currently updating the user interface of the android application and at the same time working on the raspberry pi with Patrick to implement voice control. Also working on the GUI of the android application to maintain an up to date look that satisfies the users, while delivering the full functionality of the application.

Patrick - Currently working on voice control for the raspberry pi. Also working with Lawrence to allow full compatibility with the Android application and the voice control system.

Problem:

Database - The web host database has expired but renewed the database with a different hosting server. We will have to migrate the data from the previous database to the new host.

Voice Control - Still researching a way to run the python codes from a script as the voice control program is based of running scripts.

Financial:

PCB - we decide to switch to a PCB to change the old setup to make it more compact as previously the setup was using a small breadboard to wire everything. We also had bought a stacking header as the prototype lab did not have any leftover from the last semesters PCB creations. The price of the header was about $16, as we purchased a pack.

Voice Control - We have to purchase a USB microphone as the Raspberry PI does not have a microphone embedded into the device. Reason for the USB version is that, it make it more compact thus to the fact that it's a small attachment compared to a sound card + 3.5mm microphone. Cost for these parts is undecided since we have not decided on the parts yet.

Week 8: Patrick Ng (Integration)

Dear Kristian,

This is our PiRover Project’s status update as of the week of March 18, 2018.

Current/Recent Progress:

Hardware

Since last status update, we had designed a PCB to make it more compact compared to before. Due to this change, we had to do some testing to make sure everything was running smoothly with this transition. The battery was affected by this transition as we had to put it in a new spot on the PiRover since the PCB took up most of the general space on the Raspberry Pi.

Meanwhile, for voice control, this step had to be delayed due to the fact that the microphone that was ordered was not working and are waiting for a replacement as soon as possible. Besides this setback, the PiRover is functionally and usable with the android application that is attached to it.

Android Application

In our android application, we have updated the register page information to avoid username duplicates. The communication of our Android application with our online database is now fully functional. However, the data the will be collected from the raspberry pi to the mobile application and straight to the database is currently in progress.

Database

The database information has been migrated to a new host site as we decide that this host would be better in the future for our project. The new host is called Hostinger and it is a SQL database that is accessible via PHP. After testing with the hardware and the application, the database is storing the user login information as a new user are being created. Things that need to be worked on or improved is password verification and hashing of the passwords that are stored.

Problems:

Voice Control – the setback would delay the testing with the actual microphone we wanted to use for the PiRover. Will do some testing using last semester setup for the microphone until the replacement arrives.

Android Application – need to fix the issue when the control gets stuck and will not respond to the user touch on the application.

Financial:

Even though the microphone needs to be replaced, it will not cost anything as the vendor is going to exchange the broken one for a new one.

From,

Patrick Ng

Week 9: Lawrence (Troubleshooting)

Dear Kristian,

This is our PiRovers Project's status update as of the week of Apirl 1, 2018.

Current/Recent Progress:

Hardware

Since the last status update, we have begun troubleshooting for the PiRover; Movement, stability, terrain, and other basic troubleshooting. We have also fixed a few minor bugs with the PiRover such as a circuitry issue with the PCB. The issue was causing the PiRover's sensor to be grounded and current was not able to pass through.

On March 24th 2018, our replacement microphone for the voice application to the PiRover has arrived. Testing for this function has begun. Unfortunately we are still having software issues causing invalid inputs/outputs.

Android Application

In our android application, we have begun more testing on the application as the application aspects is meeting its final stage. Although we are still redesigning the application, we believe the app is working the way we have imagined.

Database

The database we have chosen has been working successfully. As we have begun testing on both application and hardware, the database is also being tested. Data is being sucessfully delivered to the database from the android application.

Link to our current database:

https://auth-db132.hostinger.com/sql.php?server=1&db=u352817062_bane&table=pirover&pos=0&token=a79514e8bf8d4521007747c306b445c8

Other Problems

The set back of the microphone as written in our previous update report, has caused us to be behind of our schedule for implementing the voice control. This has delayed; implementation, testing, and user easy to use application for the voice control. The android application forwards, backwards, left and right buttons, can be adjusted to be pressed at the same time. The issue it's causing now is, if two movements are addressed simutanously it will cause the application to get stuck/unresponsive.

Financial

The microphone has arrived safely and is working well compared to the old microphone delivered. We have not paid any extra fees for the substitution.

From,

Lawrence Puig

Week 11: Patrick (Voice Control Issue)

Dear Kristian,

Sorry for the delay, as I forgot to email this to you.

Over the course of the past several weeks, I had been testing the voice control integration software that was going to be used on the PiRover. The software was called PiAUISuite.

During this testing phase, I ran into an issue where the software was not picking up the speech from the microphone. At first, I thought it was the microphone but the microphone was just replaced with a new one since the original one was broken when received. After doing a microphone test with another application (Alexa from last semester class), it was working perfectly fine.

I was confused on why this was happening as I followed the correct step from the creator on how to install this software and enable voice recognition. Managed to do a full reinstall to make sure nothing went wrong during the installation stage, still no luck.

I decided to research the issue to find out that the software has not been updated for some time and that Google decided to stop the TTS service that is used in this software. Dues to this, the software wasn't going to work and it would require me to find an alternative.

I looked at other voice control software like Jasper/Alexa, but it wasn't something that would integrate easily with our rover within a short period of time. So we decided to abandon it in the meantime as we couldn't risk it as the PiRover is fully functional without it. This idea would be put into future consideration since it was possible to do it but not with the time constraint we have left in the semester. I will link the forum that people has posted about this issue as a reference.

PiAUISuite was supposed to run scripts that were included in the configuration file. For example: in our case, it would have a keyword so when the program is running and pick up that keyword from the microphone and it would just run the script or commands linked which would run the code for the PiRover.

Reference: https://github.com/StevenHickson/PiAUISuite/issues/56

From Patrick Ng

4.1 Build Instructions

4.1.1 Mechanical Assembly

- Mount the laser cut chassis all together (use glue gun or tape for stability).

- Screw the micro servos into the side chassis.

- Place the Raspberry Pi on top of the chassis.

- Carefully mount the power bank on top of the Raspberry Pi or the bottom (depending on the size of the powerbank).

- Mount the sensor with its breadboard on the Raspberry Pi (preferably on the USB ports).

4.1.2 Wiring

- Simply follow the System Diagram provided in the Illustration list.

- Follow correct GPIO pins in the diagram provided.

- Refer to the code provided in the Technical References, subheading servo2.py code, for GPIO inputs.

4.1.3 Power Up

Powering up the Raspberry Pi.

- Plug in the powerbank to the Raspberry Pi and turn it on.

- Install/Updated the OS if have not already done so.
- Plug in the Raspberry Pi to any available monitor with an HDMI cable.
- Raspberry Pi Logo will appear and shortly redirect you to the desktop if ran correctly.
- Connect the Raspberry Pi to the internet using an Ethernet cable or WiFi. This allows the user to control/send commands to the Raspberry Pi remotely.

4.1.4 Unit Testing

- Use SSH, VNC, or Online VNC with an account (Preferably) for remote connection.

- Once the Raspberry Pi is setup, simply open the terminal and run the code provided under technical references in servo2.py

- W,A,S,D to control the Raspberry Pi, if code runs correctly.

4.1.5 Production Testing

- Use one of the Auto1.py code under the Technical reference to test the autonomous mode using the sensor.

4.2 Application Testing/Debugging

Test Case 1

1. General User
2. Register Account
3. No initial Conditions
4. Test Script:
    a. User opens Application from Phone
    b. Splash Activity for Application then Displays Login Page
    c. User Selects Register Here
        i. User Enters Full Name (la)
        ii. User Enters username (lap)
        iii. User Enters password (puig)
        iv. User reconfirms password (puig)

v. User Selects Register Button
d. Registered lap

Test Case 2

1. General User (lap)
2. Login as User (lap)
3. No initial Conditions
4. Test Script:
   a. User opens applicaton from Phone
   b. Splash Activity for Application the Displays Login Page
      i. User Enters username: lap
      ii. User Enters password: puig
      iii. User selects Login
   c. Main Menu Displayed

Test Case 3

1. General User (lap)
2. Settings
3. Test Case 2
4. Test Script:
   a. User Drags Navigation Drawer Out
   b. Selects Settings from Navigation Drawer
      i. User Adjust Screen Brightness by using bar
      ii. User toggles Mute/Vibrate Button
      iii. User Selects Save
   c. Displays Toast “Saved”.

Test Case 4

1. General User (lap)
2. Automatic Mode
3. Test Case 2
4. Test Script:
   a. User Selects Automatic Button
      I. User Selects CONNECT Button
      II. User Selects AUTOMAP Button
   b. User Turning off Automatic Button
      i. User Selects STOP Button
      ii. User Selects DISCONNECT Button
   c. User Navigates Back to the Main Page with Android back Button

Test Case 5

1. General User (lap)
2. Manual Mode
3. Test Case 2
4. Test Script:
   a. User Selects Manual Button

   b. User Selects CONNECT Button
      i. User touches Forward Button
      ii. User touches Back Button
      iii. User touches Left Button
      iv. User touches Right Button
   c. User Selects DISCONNECT Button

Test Case 6

1. General User (lap)
2. Developers on Github
3. Test Case 2
4. Test Script:
   a. User Selects Developers Floating Action Button
      i. User Selects Chris' Github
      ii. User Selects Lawrence's Github
      iii. User Selects Heakeme's Github
   b. User Deselects Developers Floating Action Button

Test Case 7

1. General User
2. Login As Guest
3. No Initial Conditions
4. Test Script:
   a. User Opens Application
   b. Splash Activity Displayed then Login Page
      i. User Selects Login as Guest
   c. Main Menu Displayed as User "Guest".

Test Case 8

1. General User (lap)
2. Logout As User (lap)
3. Test Case 3
4. Test Script:
   a. User Selects Android Back Button
      i. User Selects Yes for "Are you logging out?" Prompt
   b. Login Page is Displayed

Test Case 9

1. General User
2. Landscape mode
3. No Initial Conditions
4. Test Script:
   a. A User Opens Application
   b. Splash Activity Displayed then Login page
   c. User Rotated Phone Device 90*(degrees).
   d. Activities are now in Landscape mode

Test Case 10

1. General User
2. Exit Application
3. No Initial Conditions
4. Test Script:
    a. User Selects Android Back Button
        i. User Selects Yes for Prompt “Are you sure you want to exit?”
    b. User is exited and sent back to Phone Home Screen

5 Conclusion

5.1 Final Report

In this final report, we introduced our project, the PiRover and how to reproduce it. With the guides and references provided, we believe this is a reproducible product that is capable of becoming more than just a simple rover.

5.2 Final Product

Our final product of the PiRover is a rover with basic movements; forward, reverse, left and right. The rover has a basic design that is well fit for flat surfaces and dry weather. Our final product of the Android Application is a simple GUI that allows the user to control the PiRover automatically or manually.

5.3 Bugs and Fixes

As the final product is reached, we still believe the PiRover and the Android Application can be modified to implement a much better experience, such as;

PiRover (Hardware)

- Introduce larger wheels
- Introduce larger chasis for better stability

PiRover (Android Application)

- Fix button guides
- Include a guideline/help function
- Multi-button press (e.g. Forward and Left)

6 Recommendations

7 Technical References

7.1 servo2.py code

```
import
RPi.GPIO
as gpio
        import time
        import sys
        import Tkinter as tk
        from sensorA import distance
        import random
        gpio.setmode(gpio.BOARD)
        gpio.setup(7, gpio.OUT)
        gpio.setup(11, gpio.OUT)
        pwm=gpio.PWM(7,50)
        pwm2=gpio.PWM(11,50)
        pwm2.start(0)
        pwm.start(0)
        def forward(tf):
        DC=1./18.*(1)+2
        DC2=1./18.*(180)+2
        pwm.ChangeDutyCycle(DC2)
        pwm2.ChangeDutyCycle(DC)
        time.sleep(tf)
        pwm.start(0)
        pwm2.start(0)
        def reverse(tf):
        DC=1./18.*(1)+2
        pwm.ChangeDutyCycle(DC)
        DC2=1./18.*(180)+2
        pwm2.ChangeDutyCycle(DC2)
        time.sleep(tf)
        pwm.start(0)
        pwm2.start(0)
        def turnLeft(tf):
        #for i in range(0,180):
        DC=1./18.*(1)+2
        pwm.ChangeDutyCycle(DC)
        pwm2.ChangeDutyCycle(DC)
        time.sleep(tf)
        pwm2.start(0)
        pwm.start(0)
```

```
#gpio.cleanup()
def turnRight(tf):
DC=1/18.*180+2
pwm.ChangeDutyCycle(DC)
pwm2.ChangeDutyCycle(DC)
time.sleep(tf)
pwm2.start(0)
pwm.start(0)
#gpio.cleanup()
def key_input(event):
print 'Key: ', event.char
key_press = event.char
s = 0.030
if key_press.lower()=='w':
forward(s)
elif key_press.lower()=='s':
reverse(s)
elif key_press.lower()=='a':
turnLeft(s)
elif key_press.lower()=='d':
turnRight(s)
command = tk.Tk()
command.bind('<KeyPress>', key_input)
command.mainloop()
```

7.2 auto1.py

```
import
RPi.GPIO
as gpio
        import time
        import sys
        import Tkinter as tk
        from bluetooth import*
        import os
        import random



        gpio.setmode(gpio.BOARD)
```

```
gpio.setup(7, gpio.OUT)
gpio.setup(11, gpio.OUT)
pwm = gpio.PWM(7,50)
pwm2 = gpio.PWM(11,50)
pwm.start(0)
pwm2.start(0)


def forward(tf):
    DC=1./18.*(1)+2
    DC2=1./18.*(180)+2
    pwm.ChangeDutyCycle(DC2)
    pwm2.ChangeDutyCycle(DC)
    time.sleep(tf)
    pwm.start(0)
    pwm2.start(0)


def reverse(tf):
    DC=1./18.*(1)+2
    DC2=1./18.*(180)+2
    pwm.ChangeDutyCycle(DC)
    pwm2.ChangeDutyCycle(DC2)
    time.sleep(tf)
    pwm.start(0)
    pwm2.start(0)


def turnLeft(tf):

    DC=1./18.*(1)+2
    pwm.ChangeDutyCycle(DC)
    pwm2.ChangeDutyCycle(DC)
    time.sleep(tf)
    pwm.start(0)
    pwm2.start(0)


def turnRight(tf):

    DC=1./18.*(180)+2
    pwm.ChangeDutyCycle(DC)
    pwm2.ChangeDutyCycle(DC)
```

```
    time.sleep(tf)
    pwm.start(0)
    pwm2.start(0)


def stop():
    time.sleep(1)
    gpio.cleanup()


def check_front():
    dist = distance()


    if dist<15:
        print('Too close,', dist)


        reverse(1)
        turnLeft(1)

        if dist < 15:
            print("Alright... that's too close. Im done", dist)
            sys.exit()


def autonomy():
    tf = 0.030
    x = random.randrange(0,4)


    if x==0:
        for y in range(30):
            check_front()
            froward(tf)


    elif x==1:
        for y in range(30):
            check_front()
            forward(tf)
    elif x==2:
        for y in range(30):
```

```
            check_front()
            forward(tf)
```

```
server_sock = BluetoothSocket( RFCOMM )
server_sock.bind(("",PORT_ANY))
server_sock.listen(1)

port = server_sock.getsockname()[1]

uuid = ("00001101-0000-1000-8000-00805f9b34fb");

advertise_service(server_sock, "SampleServer",
        service_id = uuid,
        service_classes = [ uuid, SERIAL_PORT_CLASS ],
        profiles = [ SERIAL_PORT_PROFILE ],
    #   protocols = [ OBEX_UUID ]
        )
print("Waiting for connection on RFCOMM channel %d" % port)

client_sock, client_info = server_sock.accept()
print("Accepted connection from ", client_info)

try:
    while True:
        data = client_sock.recv(1024)
        sleep = 0.030

        if data =='w':
            print("moving forward")
            forward(sleep)
```

```
        elif data == 's':
            print("moving backwards")
            reverse(sleep)


        elif data == 'a':
            print("turning left")
            turnLeft(sleep)


        elif data == 'd':
            print("turning right")
            turnRight(sleep)


        elif data == 'automap':
            print("auto mapping")
            for z in range(10):
                autonomy()



        elif data == 'stop':
            print("stopping")
            stop()



except IOError:
    pass


print("Disconnected")


client_sock.close()
server_sock.close()
print("Done")
```

8 Appendices