Spring 4-27-2018

IoT Voice, Gesture & Application Control System: Proof of Concept Implementation

Deja Jackson

Zoe Cesar

KENNESAW STATE UNIVERSITY

IoT Voice & Gesture & Application Control System: Proof-of-Concept Implementation

Honors Capstone Document

Deja Tyla Jackson & Zoe Cesar

4/9/2018

Revision History

Date	Version	Description	Author
4/9/2018	1.0	Initial Honors Document	Deja Tyla Jackson, & Zoe Cesar

Table of Contents

.

Honors Capstone

1. Introduction

1.1 Purpose

The purpose of this Honors Capstone document is to provide a detailed overview of the project, design elements and classes.

1.2 Scope

This Honors Capstone Document will be used by the team to illustrate the development of the IoT Android Application. This document highlights a timeline of when activities were completed, includes class tables and diagrams of the application and also includes mockups and the final prototype of the app as well. The time breakdown of our project is described in the Project Schedule below.

1.3 Definitions, Acronyms and Abbreviations

IoT-(Internet of Things) - a term used to describe devices that are connected over the same wireless network that communicate with each other and exchange data

Robot (Raspberry Pi 3) - a car that consists of a Raspberry Pi 3 B as the brains, a L298N Dual h-bridge motor controller, four 18650 batteries for power and a Makerfire 4-wheel chassis.

Microcontroller- a small computer on one integrated circuit that can contains a CPU, memory, and has dedicated pins that allow the board to interact with different types of sensors (motion, light, etc.).

Initial Raspberry Pi 3 - the Raspberry Pi device in this system used for Voice Recognition and Machine Learning

App - is an abbreviation of the word Application. For the purpose of this document App refers to an Android Application.

1.4 Overview

This Document contains the following:

Project Overview - This section provides an overview of the project, including the purpose, scope, assumptions and project deliverables.

Project Organization – This section outlines the roles and responsibilities of each contributor

Project Schedule – This section outlines the schedule and timeline that was used to ensure project completion.

Design Overview – This section illustrated the User Interface of the Application

Requirements - This section discusses the performance, functional and non-functional requirements of the system.

Analysis Classes – This section outlines the classes involved in the project.

Honors Reflection - This section discusses the Honors Impact on each individual and the Foundations of Learning they believe are involved in the project.

2. Project Overview

2.1 Project purpose, scope and objectives

This project is designed as a part of a larger ongoing research project at Kennesaw State University called “Internet-of-Things based Smart Classroom Environment” undertaken by the Wireless Mobile Computing research group of the Department of Computer Science.

The purpose of this project is to build upon the senior project aspect of this project for the honors component. For senior project, our group is working on a project involving ongoing research for an “Internet-of-Things based Smart Classroom Environment”. The project is entitled “Motion and Audio based Control System: Proof-of-Concept Implementation on Robotics via Internet-of-Things (IoT) Technologies. This project involves the use of a Raspberry Pi 3 and SenseHAT used to command a robot to execute motions through means of a cloud server, and IoT technologies. More specifically, the microcontroller will be trained to recognize a set of 5 pre-defined gestures, and these gestures will be stored on the cloud server. The robot will retrieve these commands from the cloud server and react according to the gesture it has recognized. This project is but a single yet powerful tool for future applications in smart classroom technologies.

For the honors aspect of this project, we plan to develop an android application to serve as another means of control for the robot and can serve as a visual middle man within the system. The honors portion of this project involves modifying the gesture recognition system from senior project, discovering how to send messages between the Raspberry Pi and the Android Phone/Application through the cloud server, building an Android Application, sending messages between the application and the robot, and creating scripts to control the robot through Bluetooth.

2.2 Assumptions and constraints

The project has a 16-week timeframe and therefore should be completed and ready to be presented by the scheduled day of the final presentation.

For this project to work successfully and as intended, it is constrained to both Bluetooth and Wi-Fi capabilities.

2.3 Project Deliverables

The following deliverables will be produced during the project:

- Project Management
    - Proposal
    - Initial Signature Form
    - Mid Progress Report and Signature Form
    -
- Implementation
    - Android Application

3. Project Organization

The responsibilities of each team member are as follows:

Name	Responsibility
Deja Tyla Jackson	Design the Application UI, Software Documentation, Poster, Modifying Robot Scripts, Android Development(Robot Controls page and Voice Control page)
Zoe Cesar	Setup GitHub, Software Documentation, Poster, Modifying Robot Scripts, Android Development(Cloud Controls page and Help Page)

4. Project Schedule

The project schedule for this project is as follows:

- **Week of Jan 14:**
  - **Due on 1/16:** Finalize Project Schedule
  - **Due on 1/21:** Finalize Project Idea and Very Clear Idea of the Project
- **Week of Jan 21:**
  - **Due on 1/28:** Finalize the Application Design and Finalize a Logo
- **Week of Jan 28:**
  - **Due on 2/4:** Learn the basics of Android and have an outline of how the application will work programmatically; Continue to design the pages
- **Week of Feb 4:**
  - **Due on 2/11:** Have the app structure mostly completed (minus the external connections- dead links or something to external connections)
- **Week of Feb 11:**
  - **Due on 2/18:** Research and Finalize a method of connecting the Robot and Application
- **Week of Feb 18:**
  - **Due on 2/25:** Continue connecting the app and the robot
- **Week of Feb 25:**
  - **Due on 3/4:** Work on finding a way to connect the microcontroller to the app and have something sent (Via Bluetooth for now, if supported), Fully Connect App and Robot
- **Week of March 4:**
  - **Due on 3/11:** Fully connect the app and the microcontroller
- **Week of March 11:**
  - **Due on 3/18:** Research cloud server implementation on the app and finalize the implementation

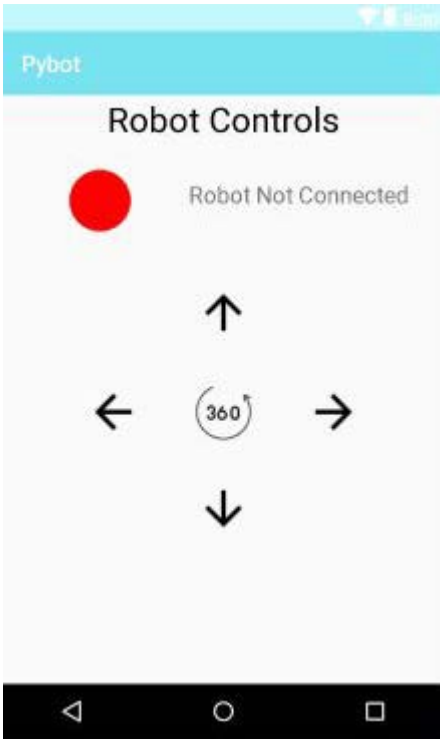
- **Week of March 18:**
    - **Due on 3/25:** Finish cloud server connection, Research Elements for the Poster
- **Week of March 25:**
    - **Due on 4/1:** Finishing touches on UI and any other of the previous, Research Elements for the Poster
- **Week of April 1:**
    - **Due on 4/8:** Research Elements for the Poster, Finish the Poster, Fix any bugs in the app and have a user test it
- **Week of April 8:**
    - **Due on 4/15:** Demonstrate and Submit Capstone Poster to Faculty Supervisor and Department Liaison (4/13)
- **Week of April 15:**
    - **Due on 4/22:** Submit Capstone Project to Honors College
- **Week of April 22:**
    - **Due on 4/27:** Capstone Graduation Celebration

5. Design Overview

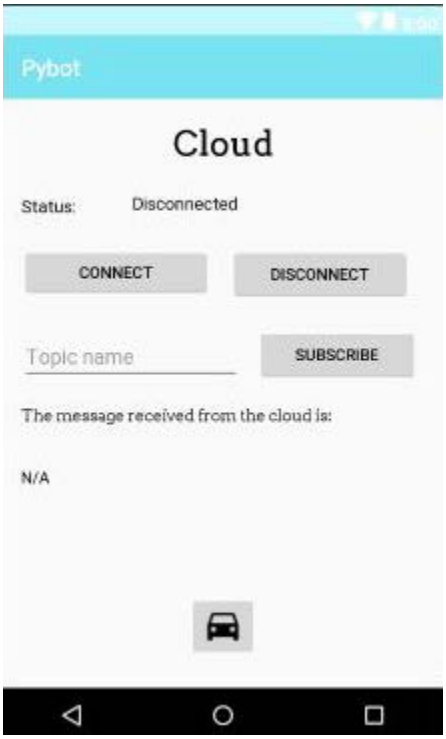
The images below illustrate the UI of the Application:

Voice Control

Robot Control



Cloud Control



FAQs

6. **Requirements**

ID: R1

Title: Cloud Latency

Description: Real-time latency of up to 5 seconds for cloud connection to app

ID: R2

Title: Bluetooth Latency

Description: Real-time latency of up to 4 seconds for the Bluetooth connection to the robot

ID: R3

Title: Usage of the Application

Description: The android application should be easy to use and understand through the use of simple and efficient User Interface**.**

ID: R4

Title: Responsiveness of the Application

Description: The android application should be responsive and not lag in performance

ID: R5

Title: Voice Control

Description: The voice recognition should be accurate and have the ability to indicate the correct voice command.

ID: R6

Title: Usage of battery

Description: The Android device requires sufficiently charged batteries and operating hardware components.

7. Classes

Class name: *HelpActivity*	
Brief description: This activity holds the FAQ page of the app	
Methods (operations)	**Method Description**

protected void onCreate(Bundle savedInstanceState)	This method is what is created when the HelpActivity is run/ selected
	
	Programming Description Language:
	
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_help);

Class name: *CloudActivity*
Brief description: This activity page is the page that holds all the information that the app receives from the cloud server. It also transfers the cloud message to the robot through bluetooth.

Attributes (fields)	**Attribute Description**
private static final String CUSTOMER_SPECIFIC_ENDPOINT = "a2zgnpn7ya2td1.iot.us-east-2.amazonaws.com";	AWS Iot CLI describe-endpoint call returns: XXXXXXXXXX.iot.<region>.amazonaws.com
private static final String COGNITO_POOL_ID = "us-east-2:be3fd889-5b08-447a-afa7-4fc749eed242";	Cognito pool ID. For this app, pool needs to be unauthenticated pool with AWS IoT permissions.
private static final String AWS_IOT_POLICY_NAME = "GesturePolicy";	Name of the AWS IoT policy to attach to a newly created certificate
private static final Regions MY_REGION = Regions.US_EAST_2;	Region of AWS IoT
private static final String KEYSTORE_NAME = "iot_keystore";	Filename of KeyStore file on the filesystem
private static final String CERTIFICATE_ID = "default";	Certificate and key aliases in the KeyStore
private static final String KEYSTORE_PASSWORD = "password";	Password for the private key in the KeyStore
TextView tvLastMessage; TextView tvStatus;	TextViews that display information from the cloud

EditText txtSubcribe; EditText txtTopic; EditText txtMessage;	Text that changes based on user activity
AWSIotClient mIotAndroidClient; AWSIotMqttManager mqttManager; String clientId; String keystorePath; String keystoreName; String keystorePassword; KeyStore clientKeyStore = null; String certificateId;	Clients that interact with the app and AWS
Button btnConnect; Button btnSubscribe; Button btnDisconnect;	Buttons that connect to the cloud
BluetoothSocket mmSocket = null;	BluetoothSocket set to null
BluetoothDevice mmDevice = null;	Bluetooth Device set to null
final byte delimiter = 33;	Sets the delimiter to a final value of 33
int readBufferPosition = 0;	Buffer position set to the initial position of 0
Methods (operations)	**Method Description**

protected void onCreate(Bundle savedInstanceState)	Method that launches when the Cloud Activity is run
	
	Programming Description Language:
	
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_cloud);
	txtSubcribe = (EditText) findViewById(R.id.txtSubcribe);
	txtTopic = (EditText) findViewById(R.id.txtTopic);
	txtMessage = (EditText) findViewById(R.id.txtMessage);

	tvLastMessage = (TextView) findViewById(R.id.tvLastMessage);
	tvStatus = (TextView) findViewById(R.id.tvStatus);
	
	btnConnect = (Button) findViewById(R.id.btnConnect);
	btnConnect.setOnClickListener(connectClick);
	btnConnect.setEnabled(false);
	
	btnSubscribe = (Button) findViewById(R.id.btnSubscribe);
	btnSubscribe.setOnClickListener(subscribeClick);
	
	btnDisconnect = (Button) findViewById(R.id.btnDisconnect);
	btnDisconnect.setOnClickListener(disconnectClick);
	
	clientId = UUID.randomUUID().toString();
	// Initialize the AWS Cognito credentials provider
	// MQTT Client
	mqttManager = new AWSIotMqttManager(clientId, CUSTOMER_SPECIFIC_ENDPOINT);
	
	// Set keepalive to 10 seconds. Will recognize disconnects more quickly but will also send
	// MQTT pings every 10 seconds.
	mqttManager.setKeepAlive(10);
	
	// Set Last Will and Testament for MQTT. On an unclean disconnect (loss of connection)
	// AWS IoT will publish this message to alert other clients.
	AWSIotMqttLastWillAndTestament lwt = new AWSIotMqttLastWillAndTestament("my/lwt/topic", "Android client lost connection", AWSIotMqttQos.QOS0);
	mqttManager.setMqttLastWillAndTestament(lwt);
	
	// IoT Client (for creation of certificate if needed)
	
	// To load cert/key from keystore on filesystem
	// load keystore from file into memory to pass on connection

new Thread(new Runnable()
@Override
public void run()
	Runs when the user requests to connect to the cloud server
	
	Programming Description Language:
	try {
	// Create a new private key and certificate. This call
	// creates both on the server and returns them to the device.
	
	// store in keystore for use in MQTT client saved as alias "default" so a new certificate isn't generated each run of this application

	// load keystore from file into memory to pass on connection
	
	clientKeyStore = AWSIotKeystoreHelper.getIotKeystore(certificateId, keystorePath, keystoreName, keystorePassword);
	
	// Attach a policy to the newly created certificate.
	// This flow assumes the policy was already created in
	// AWS IoT and we are now just attaching it to the certificate.
	AttachPrincipalPolicyRequest policyAttachRequest = new AttachPrincipalPolicyRequest();
	policyAttachRequest.setPolicyName(AWS_IOT_POLICY_NAME);
	policyAttachRequest.setPrincipal(createKeysAndCertificateResult.getCertificateArn());
	mIotAndroidClient.attachPrincipalPolicy(policyAttachRequest);
	
	runOnUiThread(new Runnable() {
	@Override
	public void run() {
	btnConnect.setEnabled(true); } });
	} catch (Exception e) {
	Log.e(LOG_TAG, "Exception occurred when generating new private key and certificate.", e);}} }).start();}}

View.OnClickListener connectClick = new View.OnClickListener() {
@Override
public void onClick(View v) {
	Listener for when the user connects to the cloud
	
	Programming Description Language:
	Log.d(LOG_TAG, "clientId = " + clientId);
	try { mqttManager.connect(clientKeyStore, new AWSIotMqttClientStatusCallback() {
	@Override
	public void onStatusChanged(final AWSIotMqttClientStatus status, final Throwable throwable) { Log.d(LOG_TAG, "Status = " + String.valueOf(status));
	
	runOnUiThread(new Runnable() {
	@Override
	public void run() {
	if (status == AWSIotMqttClientStatus.Connecting) {
	tvStatus.setText("Connecting...");
	} else if (status == AWSIotMqttClientStatus.Connected) {
	tvStatus.setText("Connected"); } else if (status == AWSIotMqttClientStatus.Reconnecting) {
	if (throwable != null) {
	Log.e(LOG_TAG, "Connection error.", throwable);}
	tvStatus.setText("Reconnecting");}
	else if (status == AWSIotMqttClientStatus.ConnectionLost) {if (throwable != null) {

	Log.e(LOG_TAG, "Connection error.", throwable);} tvStatus.setText("Disconnected"); } else { tvStatus.setText("Disconnected");}}});}});} catch (final Exception e) { Log.e(LOG_TAG, "Connection error.", e);tvStatus.setText("Error! " + e.getMessage());}}};

View.OnClickListener subscribeClick = new View.OnClickListener() {
@Override
public void onClick(View v)
	Listener for when the user subscribes to the cloud
	
	Programming Description Language:
	
	final String topic = txtSubcribe.getText().toString();
	
	Log.d(LOG_TAG, "topic = " + topic);
	try { mqttManager.subscribeToTopic(topic, AWSIotMqttQos.QOS0, new AWSIotMqttNewMessageCallback() {
	
	@Override
	public void onMessageArrived(final String topic, final byte[] data) { runOnUiThread(new Runnable() {
	@Override
	public void run() {try {String message = new String(data, "UTF-8"); Log.d(LOG_TAG, "Message arrived:");
	Log.d(LOG_TAG, " Topic: " + topic); Log.d(LOG_TAG, " Message: " + message);
	tvLastMessage.setText(message);}
	catch (UnsupportedEncodingException e)
	{ Log.e(LOG_TAG, "Message encoding error.", e);}}});}}); } catch (Exception e) {Log.e(LOG_TAG, "Subscription error.", e);}}};

View.OnClickListener disconnectClick = new View.OnClickListener() { @Override public void onClick(View v)	Listener for when the user disconnects from the cloud **Programming Language Description:** try {mqttManager.disconnect(); } catch (Exception e) {Log.e(LOG_TAG, "Disconnect error.", e); }}};}

sendBtMsg(String mes)	Sends the bluetooth message to the Raspberry Pi
	
	Programming Description Language:
	UUID uuid = UUID.fromString("94f39d29-7d6d-437d-973b-fba39e49d4ee"); //Standard SerialPortService ID
	try {
	mmSocket = mmDevice.createRfcommSocketToServiceRecord(uuid);
	if (!mmSocket.isConnected()){

	mmSocket.connect();
	}
	String msg = mes;
	OutputStream mmOutputStream = mmSocket.getOutputStream();
	mmOutputStream.write(msg.getBytes()); //Sends Messages
	} catch (IOException e) {
	e.printStackTrace();
	}

onCreateOptionsMenu (Menu menu)	Creates menu to navigate to different pages of the application. **Programming Description Language:** getMenuInflater().inflate(R.menu.menu_main, menu); return true;

onOptionsItemSelected(MenuItem item)	Based on the id of the selected menu item, the user will be redirected to the appropriate page of the application.
	
	Programming Description Language:
	int id = item.getItemId();
	
	//noinspection SimplifiableIfStatement
	if (id == R.id.main) {
	//return true;
	//TODO: Make this work to redirect
	//startActivity(new Intent(CloudActivity.this, MainActivity.class));
	
	}
	else if(id == R.id.voice){
	startActivity(new Intent(CloudActivity.this, VoiceActivity.class));
	//return true;
	}
	else if(id == R.id.cloud)
	{
	startActivity(new Intent(CloudActivity.this, CloudActivity.class));
	//return true;
	}
	else{
	startActivity(new Intent(CloudActivity.this, HelpActivity.class));
	//return true;
	}
	return super.onOptionsItemSelected(item);

sendCommandRight()	Creates a Toast to indicate that the Pybot is moving as indicated.
	
	Programming Description Language:
	Toast toast=Toast.makeText(this, "PyBot is turning right", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommandLeft()	Creates a Toast to indicate that the Pybot is moving as indicated.
	
	Programming Description Language:
	Toast toast=Toast.makeText(this, "PyBot is turning left", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommandForward()	Creates a Toast to indicate that the Pybot is moving as indicated.
	Programming Description Language:
	Toast toast=Toast.makeText(this, "PyBot is moving forwards", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommandBackwards()	Creates a Toast to indicate that the Pybot is moving as indicated.
	Programming Description Language:
	
	Toast toast=Toast.makeText(this, "PyBot is moving backwards", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommand360()	Creates a Toast to indicate that the Pybot is moving as indicated.
	Programming Description Language:
	Toast toast=Toast.makeText(this, "PyBot is turning 360 degrees", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

Class name: *VoiceActivity*

Brief description: This class allows the user to speak and translates the command into text and displays it on the screen. The user is able to press the robot button to send the bluetooth message to the robot.	
Attributes (fields)	**Attribute Description**
private TextView voiceOutputText;	TextView for the Speech Output
private ImageButton voiceBt;	ImageButton to initiate recording voice when pressed
private ImageButton robotBt;	ImageButton to iniaite a response from the robot when pressed
BluetoothSocket mmSocket = null;	BluetoothSocket set to null
BluetoothDevice mmDevice = null;	Bluetooth Device set to null
private static final int REQ_CODE_SPEECH_INPUT = 100;	Final Variable for the Speech Input
final byte delimiter = 33;	Sets the delimiter to a final value of 33
int readBufferPosition = 0;	Buffer position set to the initial position of 0
Methods (operations)	**Method Description**

onCreate(Bundle savedInstanceState)	Establishes the buttons, button events and the bluetooth pairing to the phone.
	
	Programming Description Language:
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_voice);
	
	voiceOutputText = (TextView) findViewById(R.id.voiceOutput);
	voiceBt = (ImageButton) findViewById(R.id.voice);
	robotBt = (ImageButton) findViewById(R.id.robot);
	
	voiceBt.setOnClickListener(new View.OnClickListener() {

	@Override
	public void onClick(View v) {
	startVoiceInput();
	}
	});
	//set the onClick to the Robot Command*/
	robotBt.setOnClickListener(new View.OnClickListener() {
	
	@Override
	public void onClick(View v) {
	sendRobotCommand();
	}
	}); //set the onClick to the Robot Command
	
	//Setup Bluetooth Connection and Make sure it is enabled
	BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
	
	//TOO much to include -- Code to establish the bluetooth and pairing

startVoiceInput()	Starts the Activity to begin the voice activity and record speech from the user.
	
	Programming Description Language:
	Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
	intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
	intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
	intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Say a Command! Recognized Commands: Spin, Left, Right, Back, Forward");
	
	try {
	startActivityForResult(intent, REQ_CODE_SPEECH_INPUT);
	} catch (ActivityNotFoundException a) {
	
	}

onActivityResult(int requestCode, int resultCode, Intent data)	Obtains the result from the voice data and sets the voiceOutput value.
	
	Programming Description Language:
	super.onActivityResult(requestCode, resultCode, data);
	
	switch (requestCode) {
	case REQ_CODE_SPEECH_INPUT: {
	if (resultCode == RESULT_OK && null != data) {
	ArrayList<String> result = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
	voiceOutputText.setText(result.get(0));
	}
	break;
	}
	
	}

sendRobotCommand()	Calls upon methods in order to create Toasts to show the motion. It creates threads to send to the robot in bluetooth.
	
	Programming Description Language:
	
	if(voiceOutputText.getText().toString().contains("left"))
	{
	Thread th = new Thread(new WThread("3"));
	th.start();
	sendCommandLeft();
	}else if(voiceOutputText.getText().toString().contains("right"))
	{
	Thread th = new Thread(new WThread("4"));
	th.start();
	sendCommandRight();
	}else if(voiceOutputText.toString().contains("spin"))
	{
	Thread th = new Thread(new WThread("5"));
	th.start();
	sendCommand360();
	}
	else if(voiceOutputText.getText().toString().contains("back"))
	{
	Thread th = new Thread(new WThread("2"));
	th.start();
	sendCommandBackwards();

	}else if(voiceOutputText.getText().toString().contains("forward")) { Thread th = new Thread(new WThread("1")); th.start(); sendCommandForward(); } else { //Command not recognized Toast toast = Toast.makeText(this, "PyBot has not received a recognized command", Toast.LENGTH_LONG); toast.show(); closeToast(toast); }
closeToast(Menu menu)	Closes the Toast after a specified delay **Programming Description Language:** Handler handler = new Handler(); handler.postDelayed(new Runnable() { @Override public void run() { toast2.cancel(); } }, 500);
onCreateOptionsMenu(Menu menu)	Creates menu to navigate to different pages of the application. **Programming Description Language:** getMenuInflater().inflate(R.menu.menu_main, menu); return true;
onOptionsItemSelected(MenuItem item)	Based on the id of the selected menu item, the user will be redirected to the appropriate page of the application. **Programming Description Language:** int id = item.getItemId(); //noinspection SimplifiableIfStatement if (id == R.id.main) { startActivity(new Intent(VoiceActivity.this, MainActivity.class)); //return true; } else if(id == R.id.voice){ startActivity(new Intent(VoiceActivity.this, VoiceActivity.class));

	//return true;
	}
	else if(id == R.id.cloud)
	{
	startActivity(new Intent(VoiceActivity.this, CloudActivity.class));
	//return true;
	}
	else{
	startActivity(new Intent(VoiceActivity.this, HelpActivity.class));
	//return true;
	}
	return super.onOptionsItemSelected(item)

sendBtMsg(String mes)	Sends the bluetooth message to the Raspberry Pi
	
	Programming Description Language:
	UUID uuid = UUID.fromString("94f39d29-7d6d-437d-973b-fba39e49d4ee");
	try {
	mmSocket = mmDevice.createRfcommSocketToServiceRecord(uuid);
	if (!mmSocket.isConnected()){
	mmSocket.connect();
	}
	String msg = mes;
	OutputStream mmOutputStream = mmSocket.getOutputStream();
	mmOutputStream.write(msg.getBytes()); //Sends Messages
	} catch (IOException e) {
	e.printStackTrace();
	}

sendCommandRight()	Creates a Toast to indicate that the Pybot is moving as indicated.
	
	Programming Description Language:
	Toast toast=Toast.makeText(this, "PyBot is turning right", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommandLeft()	Creates a Toast to indicate that the Pybot is moving as indicated. **Programming Description Language:** Toast toast=Toast.makeText(this, "PyBot is turning left",

	Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommandForward()	Creates a Toast to indicate that the Pybot is moving as indicated.
	Programming Description Language:
	Toast toast=Toast.makeText(this, "PyBot is moving forwards", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommandBackwards()	Creates a Toast to indicate that the Pybot is moving as indicated.
	Programming Description Language:
	
	Toast toast=Toast.makeText(this, "PyBot is moving backwards", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommand360()	Creates a Toast to indicate that the Pybot is moving as indicated.
	Programming Description Language:
	Toast toast=Toast.makeText(this, "PyBot is turning 360 degrees", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

Class name: *MainActivity*	
Brief description: This class allows the user to control the robot through pressing the buttons on the controller displayed in the xml page and sending a Bluetooth message to the robot.	
Attributes (fields)	**Attribute Description**
BluetoothSocket mmSocket = null;	BluetoothSocket set to null
BluetoothDevice mmDevice = null;	BluetoothDevice set to null

final byte delimiter = 33;	Sets the delimiter to a final value of 33
int readBufferPosition = 0;	Buffer position set to the initial position of 0
boolean connectionStatus = false;	Boolean Value to indicate if the bluetooth is connected
private TextView rConnection;	TextView to display text if the robot is connected
Methods (operations)	**Method Description**

onCreate(Bundle savedInstanceState)	Handles the onCreate of the Page and handles the Thread to send the bluetooth message
	
	Programming Description Language:
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_main);
	
	final Handler handler = new Handler();
	
	//Setup Bluetooth Connection and Make sure it is enabled
	BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
	
	//TOO much to include -- code to create the Thread to handle sending the Bluetooth Message so the Application doesn't crash
	//TOO much to include -- on click events for all of the buttons

sendBtMsg(String mes)	Sends the bluetooth message to the Raspberry Pi
	
	Programming Description Language:
	UUID uuid = UUID.fromString("94f39d29-7d6d-437d-973b-fba39e49d4ee"); //Standard SerialPortService ID
	try {
	mmSocket = mmDevice.createRfcommSocketToServiceRecord(uuid);
	if (!mmSocket.isConnected()){
	mmSocket.connect();
	}

	String msg = mes;
	OutputStream mmOutputStream = mmSocket.getOutputStream();
	mmOutputStream.write(msg.getBytes()); //Sends Messages
	} catch (IOException e) {
	e.printStackTrace();
	}

sendCommandRight()	Creates a Toast to indicate that the Pybot is moving as indicated.
	
	Programming Description Language:
	Toast toast=Toast.makeText(this, "PyBot is turning right", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommandLeft()	Creates a Toast to indicate that the Pybot is moving as indicated.
	
	Programming Description Language:
	Toast toast=Toast.makeText(this, "PyBot is turning left", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommandForward()	Creates a Toast to indicate that the Pybot is moving as indicated.
	Programming Description Language:
	Toast toast=Toast.makeText(this, "PyBot is moving forwards", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommandBackwards()	Creates a Toast to indicate that the Pybot is moving as indicated.
	Programming Description Language:
	
	Toast toast=Toast.makeText(this, "PyBot is moving backwards", Toast.LENGTH_LONG);
	toast.show();
	closeToast(toast);

sendCommand360()	Creates a Toast to indicate that the Pybot is moving as indicated. **Programming Description Language:** Toast toast=Toast.makeText(this, "PyBot is turning 360 degrees", Toast.LENGTH_LONG);

	toast.show(); closeToast(toast);

closeToast(Menu menu)	Closes the Toast after a specified delay
	
	Programming Description Language:
	Handler handler = new Handler();
	handler.postDelayed(new Runnable() {
	@Override
	public void run() {
	toast2.cancel();
	}
	}, 500);

onCreateOptionsMenu(Menu menu)	Creates menu to navigate to different pages of the application. **Programming Description Language:** getMenuInflater().inflate(R.menu.menu_main, menu); return true;

onOptionsItemSelected(MenuItem item)	Based on the id of the selected menu item, the user will be redirected to the appropriate page of the application.
	
	Programming Description Language:
	int id = item.getItemId();
	
	//noinspection SimplifiableIfStatement
	if (id == R.id.main) {
	//return true;
	//TODO: Make this work to redirect
	//startActivity(new Intent(MainActivity.this, MainActivity.class));
	
	}
	else if(id == R.id.voice){
	startActivity(new Intent(MainActivity.this, VoiceActivity.class));
	//return true;
	}
	else if(id == R.id.cloud)
	{
	startActivity(new Intent(MainActivity.this, CloudActivity.class));
	//return true;
	}
	else{
	startActivity(new Intent(MainActivity.this,

	HelpActivity.class));
	//return true;
	}
	return super.onOptionsItemSelected(item);

8. Honors Reflection

8.1 Impact and Foundations of Learning – Zoe Cesar

This project will move me in the direction of my goals for the future since machine learning is a hot research field in Computer Science. By learning these techniques, I will be able to use them in a wide variety of tasks that deal with data analysis. The skills I utilize in this project will demonstrate my skills in leadership, critical thinking, information fluency, effective communication, and creativity and innovation. As part of a team, communicating effectively is key to having a successful group dynamic, as well being able to lead when leadership is needed. I will use critical thinking when trying to solve problems along the way, such as connecting the app to the cloud. Information fluency will come with the research and being able to explain what I learned clearly and concisely. Lastly, as this project involves UI design of an app as well as app functionality, creativity will be used in order to give the user a great UX experience with the app.

8.2 Impact and Foundations of Learning – Deja Tyla Jackson

This project aligns with my future because I aspire to be a Computer Scientist within the domain of cybersecurity. Cybersecurity is an ever growing field, because many companies are not taking effective steps to protect their data and information. IoT is renown today as many of the top companies such as Samsung and Amazon take advantage of the ability of using smart devices that can assist humans in everyday tasks. For example, the IoT permits users to track their refrigerator contents from an application on their mobile device, or to even change the temperature of their homes. While these technologies may seem resourceful, they generally lack efficient security protocols and are vulnerable to attacks. As a future security specialist, I aspire to have expertise in protecting users and ensuring these devices meet the adequate security protocols.

In my opinion, this project aligns with the “Critical thinking” Honors Foundation of Learning as well as the “Creativity and Innovation” foundation. Critical thinking is a very complex concept, and involves unbiased analysis and evaluations of facts in order to come to a conclusion. Well-designed projects involve taking skills and knowledge, to create a product. This project requires me to take the skills I have learned in User Interface Design to design a UI, and the skills I learned in Programming principles to develop the application, as well as learn new skills to connect and use web servers and Bluetooth communication for IOT devices. Creativity and Innovation is also a part of this honors project because I am able to have ingenuity in the development and design of my idea. Therefore, this project can be categorized as honors because it goes above and beyond what I am already doing in Senior Project to perfect an idea in a creative manner. Additionally, I am applying many concepts I have learned over a variety of courses, research and internship experiences to create a tangible software product.