USB/Bluetooth Media Controller
Final Report

40056761
SET09118
Edinburgh Napier University

22-04-2016

1 Abstract

This document is the final report and a conclusion to the ”USB/Bluetooth Media Controller” project started in February [Appendix D] [Appendix E]. The purpose of this project was to research, construct and evaluate a micro-controller based system and mobile application, that would allow a user to communicate with a media device via more intuitive methods.

Contents

List of Figures

List of Tables

Listings

2 Introduction

2.1 Background and Rational

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

2.2 Aims and Deliverables

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

3 Design

3.1 Physical System Design

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Table 1: A usage table of the HID commands utilized by the media-controller

Usage ID	Usage Name	Usage Type
0xCD	Play/Pause	OSC
0xB0	Play	OOC
0xB1	Pause	OOC
0xB3	Fast Forward	OOC
0xB4	Rewind	OOC
0xB5	Scan Next Track	OSC
0xB6	Scan Previous Track	OSC
0xB7	Stop	OSC
0xE2	Mute	OOC
0xE9	Volume Increment	RTC
0xEA	Volume Decrement	RTC

3.2 Mobile Application Design

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

4 Implementation

4.1 Micro-controller Implementation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

4.2 Mobile Application Implementation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

5 Results

5.1 Achievements

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

5.2 Recommendations

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer

id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

6 Future Work

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

7 Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

8 Evaluation of Achievement

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Appendices

A Arduino Media-Controller Source Code

```
#include <Bounce2.h>
#include <Media.h>
#include <SoftwareSerial.h>

SoftwareSerial mySerial(8, 9); // RX, TX

char myChar;

//declare array of buttons
const int button[] = {3, 4, 5, 6, 7};
//declare array of debouncers
Bounce debounce[sizeof(button)];
//declare interval for debouncers
const int del = 5;
const int volumeDelay = 50;

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);

  //define buttons/bouncers attach a bouncer to each button
  for (int i = 0; i < sizeof(button); i++) {
    pinMode(button[i], INPUT);
    debounce[i] = Bounce();
    debounce[i].attach(button[i]);
    debounce[i].interval(del);
  }
}

void checkDebounce() {
  //update each debouncer i.e. check for presses
  for (int i = 0; i < sizeof(button); i ++) {
    debounce[i].update();
  }
}

void play() {
  Media.play();
  Media.clear();
}
void prev() {
  Media.previous();
  Media.clear();
}
void next() {
  Media.next();
  Media.clear();

}
void up() {
  Media.increase();
  Media.clear();
}
void down() {
  Media.decrease();
  Media.clear();
}

void loop() {

//  checkDebounce();
//
//  if (debounce[0].fell() == HIGH) {
//    prev();
//  }
//
//  if (debounce[1].fell() == HIGH) {
//    play();
//  }
//
//  if (debounce[2].fell() == HIGH) {
//    next();
//  }
//
//  if (debounce[3].read() == HIGH) {
```

```
76 //    up();
77 //    delay(volumeDelay);
78 //  }
79 //
80 //  if (debounce[4].read() == HIGH) {
81 //    down();
82 //    delay(volumeDelay);
83 //  }
84
85   while (mySerial.available()) {
86     myChar = mySerial.read();
87     Serial.print(myChar);
88
89     switch (myChar) {
90       case't':
91         play();
92         break;
93       case'p':
94         prev();
95         break;
96       case'n':
97         next();
98         break;
99       case'u':
100        up();
101        break;
102      case'd':
103        down();
104        break;
105      default:
106        break;
107    }
108    myChar = ' ';
109  }
110
111 }
```

Listing 1: mediaController_debounce.ino

B Arduino Media Key Library

```
#ifndef MEDIA_h
#define MEDIA_h

#include "HID.h"

#if !defined(_USING_HID)

#warning "Using legacy HID core (non pluggable)"

#else
//================================================================================
//================================================================================
//  Media

#define MEDIA_CLEAR 0
#define VOLUME_UP 1
#define VOLUME_DOWN 2
#define VOLUME_MUTE 4
#define MEDIA_PLAY 8
#define MEDIA_PAUSE 16
#define MEDIA_STOP 32
#define MEDIA_NEXT 64
#define MEDIA_PREVIOUS 128
#define MEDIA_FAST_FORWARD 256
#define MEDIA_REWIND 512

class Media_
{
private:
public:
    Media_(void);
    void begin(void);
    void end(void);

    // Volume
    void increase(void);
    void decrease(void);
    void mute(void);

    // Playback
    void play(void);
    void pause(void);
    void stop(void);

    // Track Controls
    void next(void);
    void previous(void);
    void forward(void);
    void rewind(void);

    // Send an empty report to prevent repeated actions
    void clear(void);
};
extern Media_ Media;

#endif
#endif
```

Media.h

```
#include "Media.h"

#if defined(_USING_HID)

static const uint8_t _hidReportDescriptor[] PROGMEM = {

    /* Cross-platform support for controls found on IR Medias */

    0x05, 0x0c,                    //   Usage Page (Consumer Devices)
    0x09, 0x01,                    //   Usage (Consumer Control)
    0xa1, 0x01,                    //   Collection (Application)
    0x85, 0x04,                    //   REPORT_ID (4)
    0x15, 0x00,                    //   Logical Minimum (0)
    0x25, 0x01,                    //   Logical Maximum (1)
    0x09, 0xe9,                    //   Usage (Volume Up)
    0x09, 0xea,                    //   Usage (Volume Down)
    0x75, 0x01,                    //   Report Size (1)
    0x95, 0x02,                    //   Report Count (2)
    0x81, 0x06,                    //   Input (Data, Variable, Relative)

    0x09, 0xe2,                    //   Usage (Mute)
    0x95, 0x01,                    //   Report Count (1)
    0x81, 0x06,                    //   Input (Data, Variable, Relative)

    0x09, 0xcd,                    //   Usage (Play)
    0x95, 0x01,                    //   Report Count (1)
    0x81, 0x06,                    //   Input (Data, Variable, Relative)

    0x09, 0xb1,                    //   Usage (Pause)
    0x95, 0x01,                    //   Report Count (1)
    0x81, 0x06,                    //   Input (Data, Variable, Relative)

    0x09, 0xb7,                    //   Usage (Stop)
    0x95, 0x01,                    //   Report Count (1)
    0x81, 0x06,                    //   Input (Data, Variable, Relative)

    0x09, 0xb5,                    //   Usage (Next)
    0x95, 0x01,                    //   Report Count (1)
    0x81, 0x06,                    //   Input (Data, Variable, Relative)

    0x09, 0xb6,                    //   Usage (Previous)
    0x95, 0x01,                    //   Report Count (1)
    0x81, 0x06,                    //   Input (Data, Variable, Relative)

    0x09, 0xb3,                    //   Usage (Fast Forward)
    0x95, 0x01,                    //   Report Count (1)
    0x81, 0x06,                    //   Input (Data, Variable, Relative)

    0x09, 0xb4,                    //   Usage (Rewind)
    0x95, 0x01,                    //   Report Count (1)
    0x81, 0x06,                    //   Input (Data, Variable, Relative)

    0x95, 0x06,                    //   Report Count (6) Number of bits remaining in byte
    0x81, 0x07,                    //   Input (Constant, Variable, Relative)
    0xc0                           //   End Collection
};

Media_::Media_(void)
{
    static HIDSubDescriptor node(_hidReportDescriptor, sizeof(_hidReportDescriptor));
    HID().AppendDescriptor(&node);
}

void Media_::begin(void)
{
}

void Media_::end(void)
{
}

void Media_::increase(void)
{
    u8 m[2];
    m[0] = VOLUME_UP;
    m[1] = 0;
    HID().SendReport(4,m,2);
}

void Media_::decrease(void)
{
    u8 m[2];
    m[0] = VOLUME_DOWN;
```

```
84      m[1] = 0;
85      HID().SendReport(4,m,2);
86  }
87
88  void Media_::mute(void)
89  {
90      u8 m[2];
91      m[0] = VOLUME_MUTE;
92      m[1] = 0;
93      HID().SendReport(4,m,2);
94  }
95
96  void Media_::play(void)
97  {
98      u8 m[2];
99      m[0] = MEDIA_PLAY;
100     m[1] = 0;
101     HID().SendReport(4,m,2);
102 }
103
104 void Media_::pause(void)
105 {
106     u8 m[2];
107     m[0] = MEDIA_PAUSE;
108     m[1] = 0;
109     HID().SendReport(4,m,2);
110 }
111
112 void Media_::stop(void)
113 {
114     u8 m[2];
115     m[0] = MEDIA_STOP;
116     m[1] = 0;
117     HID().SendReport(4,m,2);
118 }
119
120 void Media_::next(void)
121 {
122     u8 m[2];
123     m[0] = MEDIA_NEXT;
124     m[1] = 0;
125     HID().SendReport(4,m,2);
126 }
127
128 void Media_::previous(void)
129 {
130     u8 m[2];
131     m[0] = MEDIA_PREVIOUS;
132     m[1] = 0;
133     HID().SendReport(4,m,2);
134 }
135
136 void Media_::forward(void)
137 {
138     u8 m[2];
139     m[0] = 0;
140     m[1] = MEDIA_FAST_FORWARD >> 8;
141     HID().SendReport(4,m,2);
142 }
143
144 void Media_::rewind(void)
145 {
146     u8 m[2];
147     m[0] = 0;
148     m[1] = MEDIA_REWIND >> 8;
149     HID().SendReport(4,m,2);
150 }
151
152 void Media_::clear(void)
153 {
154     u8 m[2];
155     m[0] = 0;
156     m[1] = 0;
157     HID().SendReport(4,m,2);
158 }
159
160 Media_ Media;
161
162 #endif
```

Media.cpp

C Android Application Source code

```
1  package uk.co.sam.mediacontroller;
2
3  import android.content.Intent;
4  import android.os.Bundle;
5  import android.os.Handler;
6  import android.support.design.widget.Snackbar;
7  import android.support.v7.app.AppCompatActivity;
8  import android.support.v7.widget.Toolbar;
9  import android.util.Log;
10 import android.view.MotionEvent;
11 import android.view.View;
12 import android.view.Menu;
13 import android.view.MenuItem;
14 import android.view.ViewGroup;
15 import android.widget.Button;
16 import android.widget.RelativeLayout;
17
18 import java.util.ArrayList;
19
20 public class MainActivity extends AppCompatActivity {
21
22     private static ArrayList<Button> mFunctionButtons;
23     private static ArrayList<Button> mOtherButtons;
24     private BluetoothHandler mBluetoothHandler;
25
26     @Override
27     protected void onCreate(Bundle savedInstanceState) {
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_main);
30         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
31         setSupportActionBar(toolbar);
32
33         final Button button0 = (Button) findViewById(R.id.main_button_0);
34         assert button0 != null;
35         button0.setOnClickListener(new View.OnClickListener() {
36             public void onClick(View v) {
37                 mBluetoothHandler.writeValue("p");
38             }
39         });
40         final Button button1 = (Button) findViewById(R.id.main_button_1);
41         assert button1 != null;
42         button1.setOnClickListener(new View.OnClickListener() {
43             public void onClick(View v) {
44                 mBluetoothHandler.writeValue("t");
45             }
46         });
47         final Button button2 = (Button) findViewById(R.id.main_button_2);
48         assert button2 != null;
49         button2.setOnClickListener(new View.OnClickListener() {
50             public void onClick(View v) {
51                 mBluetoothHandler.writeValue("n");
52             }
53         });
54         final Button button3 = (Button) findViewById(R.id.main_button_3);
55         assert button3 != null;
56         button3.setOnTouchListener(new View.OnTouchListener() {
57             private Handler mHandler;
58             @Override public boolean onTouch(View v, MotionEvent event) {
59                 switch(event.getAction()) {
60                     case MotionEvent.ACTION_DOWN:
61                         if (mHandler != null) return true;
62                         mHandler = new Handler();
63                         mHandler.postDelayed(mAction, 500);
64                         break;
65                     case MotionEvent.ACTION_UP:
66                         if (mHandler == null) return true;
67                         mHandler.removeCallbacks(mAction);
68                         mHandler = null;
69                         break;
70                 }
71                 return false;
72             }
73             Runnable mAction = new Runnable() {
74                 @Override public void run() {
75                     mBluetoothHandler.writeValue("u");
76                     mHandler.postDelayed(this, 50);
77                 }
78             };
79         });
80         final Button button4 = (Button) findViewById(R.id.main_button_4);
```

```
81          assert button4 != null;
82          button4.setOnTouchListener(new View.OnTouchListener() {
83              private Handler mHandler;
84              @Override public boolean onTouch(View v, MotionEvent event) {
85                  switch(event.getAction()) {
86                      case MotionEvent.ACTION_DOWN:
87                          if (mHandler != null) return true;
88                          mHandler = new Handler();
89                          mHandler.postDelayed(mAction, 500);
90                          break;
91                      case MotionEvent.ACTION_UP:
92                          if (mHandler == null) return true;
93                          mHandler.removeCallbacks(mAction);
94                          mHandler = null;
95                          break;
96                  }
97                  return false;
98              }
99              Runnable mAction = new Runnable() {
100                 @Override public void run() {
101                     mBluetoothHandler.writeValue("d");
102                     mHandler.postDelayed(this, 50);
103                 }
104             };
105         });
106
107         mFunctionButtons = new ArrayList<>();
108         mOtherButtons = new ArrayList<>();
109         ViewGroup layout = (ViewGroup) findViewById(R.id.main_button_layout);
110         assert layout != null;
111         int childCount = layout.getChildCount();
112         for (int i = 0; i < childCount; i++) {
113             View child = layout.getChildAt(i);
114             if (child instanceof Button) {
115                 mFunctionButtons.add((Button) child);
116             }
117         }
118
119         RelativeLayout mainLayout = (RelativeLayout) findViewById(R.id.main_layout);
120         final Button buttonConnect = new Button(this);
121         buttonConnect.setText(R.string.main_activity_connect_button);
122         assert mainLayout != null;
123         mainLayout.addView(buttonConnect);
124         RelativeLayout.LayoutParams layoutParams =
125                 (RelativeLayout.LayoutParams) buttonConnect.getLayoutParams();
126         layoutParams.addRule(RelativeLayout.CENTER_IN_PARENT, RelativeLayout.TRUE);
127         buttonConnect.setLayoutParams(layoutParams);
128         buttonConnect.setOnClickListener(new View.OnClickListener() {
129             public void onClick(View v) {
130                 if (mBluetoothHandler.isEnabled()) {
131                     mBluetoothHandler.showPairedDialog();
132                 } else {
133                     mBluetoothHandler.enableBluetooth();
134                 }
135             }
136         });
137         mOtherButtons.add(buttonConnect);
138         hideButtons();
139
140         mBluetoothHandler = new BluetoothHandler(this, toolbar);
141     }
142
143     public static void hideButtons() {
144         for (Button button : mFunctionButtons) {
145             button.setVisibility(View.GONE);
146         }
147         for (Button button : mOtherButtons) {
148             button.setVisibility(View.VISIBLE);
149         }
150
151     }
152
153     public static void showButtons() {
154         for (Button button : mFunctionButtons) {
155             button.setVisibility(View.VISIBLE);
156         }
157         for (Button button : mOtherButtons) {
158             button.setVisibility(View.GONE);
159         }
160     }
161
162     @Override
163     public boolean onCreateOptionsMenu(Menu menu) {
```

```
164         // Inflate the menu; this adds items to the action bar if it is present.
165         getMenuInflater().inflate(R.menu.menu_main, menu);
166         return true;
167     }
168 
169     @Override
170     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
171         if (requestCode == 1) {
172             mBluetoothHandler.onActivityResult(requestCode, resultCode);
173         }
174         super.onActivityResult(requestCode, resultCode, data);
175     }
176 
177     @Override
178     public boolean onOptionsItemSelected(MenuItem item) {
179         // Handle action bar item clicks here. The action bar will
180         // automatically handle clicks on the Home/Up button, so long
181         // as you specify a parent activity in AndroidManifest.xml.
182         int id = item.getItemId();
183         switch (id) {
184             case R.id.action_disconnect:
185                     mBluetoothHandler.disconnectDevice();
186             case R.id.action_settings:
187         }
188         return super.onOptionsItemSelected(item);
189     }
190 
191     @Override
192     public void onPause() {
193         super.onPause();  // Always call the superclass method first
194         mBluetoothHandler.disconnectDevice();
195     }
196 
197     @Override
198     public void onResume() {
199         super.onResume();  // Always call the superclass method first
200         hideButtons();
201     }
202 
203 }
```

MainActivity.java

```java
package uk.co.sam.mediacontroller;

import android.app.Activity;
import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.DialogInterface;
import android.content.Intent;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AlertDialog;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

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

/**
 * Created by Sam Dixon on 22/03/2016.
 */
public class BluetoothHandler {

    private static final int REQUEST_ENABLE_BT = 1;
    private BluetoothAdapter mBluetoothAdapter;
    private View mView;
    private Activity mActivity;
    private BluetoothDevice mDevice;
    private BluetoothSocket mSocket;
    private OutputStream mOutput;
    private static final UUID MY_UUID = UUID
            .fromString("00001101-0000-1000-8000-00805f9b34fb");
    private ProgressDialog progress;


    public BluetoothHandler(Activity activity, View view) {
        this.mView = view;
        this.mActivity = activity;
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if (mBluetoothAdapter == null) {
            Snackbar.make(this.mView, "Bluetooth unavailable on this device.", Snackbar.LENGTH_LONG).↩
     show();

        }
    }

    public void enableBluetooth() {
        if (!isEnabled()) {
            Intent btOnIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            mActivity.startActivityForResult(btOnIntent, REQUEST_ENABLE_BT);
        } else {
            //do nothing
        }
    }

    public boolean isEnabled() {
        return mBluetoothAdapter.isEnabled();
    }

    protected void onActivityResult(int requestCode, int resultCode) {
        if (requestCode == REQUEST_ENABLE_BT) {
            if (resultCode != Activity.RESULT_OK) {
                Snackbar.make(mView, "Could not enable Bluetooth", Snackbar.LENGTH_LONG).show();
            } else {
                showPairedDialog();
            }
        }
    }

    public void showPairedDialog() {
        enableBluetooth();
        //define various dialog attributes
        final AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(mActivity);
        final LayoutInflater inflater = mActivity.getLayoutInflater();
        final View dialogLayout =
                inflater.inflate(R.layout.paired_dialog,
```

```
                    (ViewGroup) mActivity.findViewById(R.id.paired_dialog_list));
        final ListView pairedListView = (ListView) dialogLayout
                .findViewById(R.id.paired_dialog_list);
        final Set<BluetoothDevice> pairedDevices = mBluetoothAdapter
                .getBondedDevices();
        final ArrayAdapter<String> btArrayAdapter =
                new ArrayAdapter<>(mActivity, android.R.layout.simple_list_item_1);

        //build dialog
        dialogBuilder.setTitle("Paired Devices");
        dialogBuilder.setView(dialogLayout);
        pairedListView.setAdapter(btArrayAdapter);
        for (BluetoothDevice device : pairedDevices) {
            btArrayAdapter.add(device.getName());
        }

        //dialog cancel button
        dialogBuilder.setNegativeButton(R.string.bluetooth_handler_paired_dialog,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int id) {
                        dialog.dismiss();
                    }
                });

        //create  dialog
        final AlertDialog ad = dialogBuilder.create();

        //clickable list
        pairedListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                String pairedName = ((TextView) view).getText().toString();
                Log.i("Clicked", pairedName);
                for (BluetoothDevice device : pairedDevices) {
                    Log.i("Checking", device.getName());
                    if (pairedName.equals(device.getName())) {
                        mDevice = device;
                        Log.i("BT Device Set", mDevice.getName());
                        ad.dismiss();
                        try {
                            connectDevice();
                            Log.d("Bluetooth", "attempting connection");
                        } catch (IOException e) {
                            e.printStackTrace();
                            Log.d("connectDevice", "failure");
                        }
                        break;
                    }
                }
            }
        });
        //display dialog
        ad.show();
    }

    void connectDevice() throws IOException {
        progress = ProgressDialog.show(mActivity, null, "Connecting to " + mDevice.getName(), true);
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    mSocket = mDevice.createRfcommSocketToServiceRecord(MY_UUID);
                    mSocket.connect();
                    mOutput = mSocket.getOutputStream();
                    progress.dismiss();
                    Snackbar.make(mView, "Connected successful", Snackbar.LENGTH_SHORT).show();

                    mActivity.runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            MainActivity.showButtons();

                        }
                    });

                } catch (IOException e) {
                    e.printStackTrace();
                    Log.d("connectDevice", "no worky");
                    progress.dismiss();

                    Snackbar snack = Snackbar.make(mView, "Connection failed", Snackbar.LENGTH_LONG);
                    snack.setAction("Retry", new View.OnClickListener() {
```

```
166                         @Override
167                         public void onClick(View v) {
168                             try {
169                                 connectDevice();
170                             } catch (IOException e1) {
171                                 e1.printStackTrace();
172                             }
173                         }
174                     });
175                     snack.show();
176                 }
177             }
178         }).start();
179     }
180 
181     public void disconnectDevice() {
182         if (mSocket != null) {
183             if (mSocket.isConnected()) {
184                 try {
185                     mOutput.close();
186                     mSocket.close();
187                     Snackbar.make(mView, "Disconnected from " + mDevice.getName(), Snackbar.↩
    LENGTH_SHORT).show();
188                     MainActivity.hideButtons();
189 
190                 } catch (IOException e) {
191                     Log.d("disconnectDevice", "failed to disconnect" + e);
192                     e.printStackTrace();
193 
194                 }
195             } else {
196                 Snackbar.make(mView, "Nothing to disconnect", Snackbar.LENGTH_SHORT).show();
197             }
198         } else {
199             Snackbar.make(mView, "Nothing to disconnect", Snackbar.LENGTH_SHORT).show();
200         }
201     }
202 
203     public void writeValue(String val) {
204         try {
205             mOutput.write(val.getBytes());
206         } catch (IOException | NullPointerException e) {
207             e.printStackTrace();
208         }
209     }
210 
211 }
```

BluetoothHandler.java

D Initial Project Proposal

USB / Bluetooth Media Controller

Initial Project Proposal

Name: Sam Dixon

Matric No: 40056761

Abstract

A Media Controller allows a user to interact with an entertainment device in a manner other than a traditional graphical interface or a keyboard and mouse. By providing physical controls a user is able to quickly alter media on a PC, without the need to change application window, or the use of a keyboard and mouse. A typical media controller allows a user to skip forward or backwards on a playlist, pause/resume media and control the volume the system. Additional features may include the ability to save and play favorited media or advance audio customization, such as bass or gain level.

In order to recreate a media controller, research in to pre-existing solitons must first take place. This will include devices such as DJ decks, USB keyboards, plug-and-play panels, etc. Once these technologies have been researched, their functionality can be reversed engineered and recreated on a smaller scale using a microcontroller. Once the physical system is fully functional a mobile application will be developed, to allow remote access to the media functions provided by the micro controller.

The goal of this project is to produce two deliverables, a microcontroller based USB media controller and a companion android application. With these systems working in tandem, the functionality of a media controller can be recreated, on a more personal, cost effective scale.

Introduction

Background

The rational for this project was simple; to provide a quick and efficient way for a user to alter media being played on a PC.

While many keyboard already offer inbuilt media keys, they can often be in hard-to-reach locations or require awkward key-combinations to use. Theses reason make a standalone media controller an attractive option to users, as the controller can be placed in an easy-to-reach location and a simple, analogue interface ensures that the correct function is always selected.

While the technologies that allow users to experience their media have grown ever more advance, the actual method of controlling said media remains relatively unchanged. The main benefits of an external controller include; it allows a user to augment their media without losing access to the keyboard and mouse, haptic feedback on button presses, simplified controls (useful for elderly/children) and access to media remotely or when the PC is locked.

Basing the system around an Arduino microcontroller allows for simple expansion of the project. For instance; a Bluetooth module will allow the media controller to be accessed remotely via a mobile application. The main benefit of this is that allows a user to control their media remotely for instance; at a party the computer can be locked to prevent undue access to the system but the host can still change the current song through the mobile application.

Aims and deliverables

The overall goal of the project is create a USB/Bluetooth media controller and mobile application.

The physical device will consist of Arduino microcontroller with an individual button for each media function, namely; previous track, next track, play/pause and mute. Volume control will also be present, either in the form of a rotary knob or two buttons; one to increase and the other decrease volume. A Bluetooth module will also be present, to allow a mobile device to connect to the system and control it remotely.

The mobile application's goal is to provide the same functionality as the hub, via a Bluetooth connection to the microprocessor. In addition to the media functions present on the hub, the application can include additional operations, such as locking the computer remotely.

In order to use the app, the user must first connect to the media control hub via Bluetooth. The user will then have access to several labelled buttons that offer the same functionality of the hub.

The goal for the appearance of the app is to keep it as simple as possible, in order to reduce the overall learning curve and keep the experience intuitive.

Design choices

Deliverables

The final deliverables of this project will be; an Arduino based media controller and Sketch, and an Android application. The controller will feature several buttons to allow physical input, and the Android application will provide the same functionality but with digital input.

Scope

In an ideal world, the media controller would work with all applications "out-of-the-box", but as there is not a unified standard for media control input, this is not possible. To compromise, the controller will be designed to work on a PC running a Windows operating system. In terms of what media

programs will be supported, Google's Play Music will be main focus, with other programs included if time allows.

On the application side of the project, only Android devices will be supported. The app is also being designed with mobile phones in mind, meaning that it will not be optimized for tablet sized screens.

Method

In order to provide the functionality required to complete this project, the micro controller needs to communicate the requested function to a media program in a manner that can be understood and carried out. The simplest way to achieve this is to have the microcontroller imitate a physical system that can already do this i.e. a USB keyboard.

Preliminary research indicates that an Arduino Leonardo board will provide the best start point for this project. The reason for the Leonardo, over the more popular Duo board is because the formers single processing capabilities, allowing for improved USB communication. The Leonardo board also typically retails at a lower price than the Duo, leading to more cost effective project.

Milestones

With this project primarily serving as learning initiative, there are serval milestones that aim to be met over the project lifecycle, namely:

- Correct wiring of a microcontroller / breadboard device.
- Using a microcontroller as a basic human interface device e.g. printing text in a document.
- Using a microcontroller to send media control signals to a PC, e.g. play/pause music.
- Develop an android application to interface with a microcontroller.

With the final goal in mind, these milestones should offer motivation and provide sufficient evidence

Resources

To complete this project, the following resources will be required:

- Android Studio (IDE) – for producing the Android application.
- Arduino Software (IDE) – for developing the Sketch for the microcontroller.
- Arduino Leonardo Starter kit – providing a microcontroller to run the system as well as the hardware required to implement all the features (breadboard, resistors, etc.).
- Arduino Bluetooth module – to allow communication between the microcontroller and a mobile device.
- Android mobile phone – for testing the system, in this case a HTC One M7 will be used for the majority of testing.
- Windows PC – for development and testing.
- Google Play Music – for testing functionality.

References

[1] Arduino.cc, "Arduino - MouseKeyboard", 2016. [Online]. Available: https://www.arduino.cc/en/Reference/MouseKeyboard. [Accessed: 18- Feb- 2016].

[2] Arduino.cc, "Arduino - ArduinoLeonardoMicro", 2016. [Online]. Available: https://www.arduino.cc/en/Guide/ArduinoLeonardoMicro. [Accessed: 18- Feb- 2016].

[3]B. Benchoff, "Turning an Arduino into a USB keyboard", *Hackaday*, 2012. [Online]. Available: http://hackaday.com/2012/06/29/turning-an-arduino-into-a-usb-keyboard/. [Accessed: 18- Feb- 2016].

[4] Instructables.com, "PC USB Media Volume Controller based on Arduino", 2016. [Online]. Available: http://www.instructables.com/id/PC-USB-Media-Volume-Controller-based-on-Arduino/. [Accessed: 18- Feb- 2016].

[5] Proto-PIC, "Proto-PIC Starter Kit for Arduino Leonardo", 2016. [Online]. Available: https://proto-pic.co.uk/proto-pic-starter-kit-for-arduino-leonardo/. [Accessed: 18- Feb- 2016].

E Interim Report

USB / Bluetooth Media Controller

Interim Report

Name: Sam Dixon

Matriculation No: 40056761

Abstract

This document is an interim report summarising the current state of the "USB / Bluetooth Media Controller". The project consists of the research, development and testing of a microcontroller based media controller and a companion mobile application. The purpose of this report is to catalogue the project's various stages of development and to describe future action.

Table of Contents

1 Introduction

1.1 Background

The rationale for this project was simple - to provide a quick and efficient way for a user to alter the way media is played on a PC.

While many keyboards already offer inbuilt media keys, they can often be in locations that are difficult to access, or they require the usage of awkward key-combinations. These reasons make a stand-alone media controller an attractive option to users, as the controller can be placed in a location that is easy to reach, and a simple analogue interface ensures that the correct function is always selected.

While the technologies that allow users to experience their media have grown ever more advanced, the actual method of controlling this media remains relatively unchanged. The main benefits of an external controller include allowing a user to augment their media without losing access to the keyboard and mouse; haptic feedback on button presses; simplified controls (something which may be particularly useful for elderly users or children); and access to media remotely, or when the PC is locked.

Basing the system around an Arduino microcontroller allows for simple expansion of the project. For instance, a Bluetooth module will allow the media controller to be accessed remotely via a mobile application. The main benefit of this is that it allows a user to control their media remotely. At a party, for instance, the host's computer could be locked to prevent undue access to the system, but the host can still change the current song through the mobile application.

1.2 Aims and Deliverables

The original goal of the project was to create a USB/Bluetooth media controller and mobile application. While this remains the ultimate goal, additional functionality in the form of Windows short cuts or hotkeys is likely to be included in the final build.

The physical device will consist of an Arduino microcontroller with an individual button for each media function. These functions will be as follows - previous track; next track; play/pause; mute. Volume control will also be present in the form of two buttons (one to increase and the other decrease volume). A Bluetooth module will also feature, to allow a mobile device to connect to the system and control it remotely.

The mobile application’s goal is to provide the same functionality as the hub, via a Bluetooth connection to the microprocessor. In addition to the media functions present on the hub, the application will include various Windows hotkeys, such as Lock PC and Log Off. At this point in time, it is currently unclear whether these functions will only be accessible via the app, or if the application will change the functionality of the physical buttons.

In order to use the app, the user must first connect to the media control hub via Bluetooth. The user will then have access to several labelled buttons that offer the same functionality of the hub.

The goal for the appearance of the app is to keep it as simple as possible, in order to reduce any potential learning curve involved in usage and keep the experience intuitive.

2 Current Status

2.1 Research and Analysis

From the preliminary research it was clear that this project was feasible, however, achieving the desired functionality without the necessary proprietary software running on the PC would make the task more complex.

By reviewing the datasheets for various microcontrollers [1] [2] and the projects others had produced with them [3], it was determined that an Arduino Leonardo had the capabilities to operate as a plug-and-play USB device. Had this not been the case, a script would have to be produced that would essentially 'listen' for serial input from the microcontroller and respond appropriately.

With the knowledge that a Leonardo could act as a suitable base for the project, further research as to how the microcontroller HID operations began. By examining the Leonardo's HID, USBAPI, Keyboard and Mouse libraries [4]a better understanding of the device's USB protocols was attained, as well as providing some base code for the bespoke Media Key library being produced.

Further understanding of how USB-HID devices operated was required to fully comprehend how the Leonardo communicates with a host PC. The official USB developers guidelines [5] provided a wealth of knowledge on the operation of USB devices, specifically HID Usage Tables [6]. In essence, a Usage Table is a list of all possible operations a USB HID device can perform. This list is sent to the PC when the device is plugged in and at any point, the device can request the host to carry out any of the functions provided.

The final step in producing a fully functional Media Key library was to acquire the HEX codes of each command that would be included in HID Usage Table. A three year-old personal blog post that had previously been dismissed for being too outdated provided a sample Usage Table for a host of different media commands [7]. The post in question had been analysed as potential solution to the project during the initial research phase. However, due to the age of the post and the fact the code was designed to interact with an IDE fourteen iterations out of date (as of time of writing), it was deemed unusable. Even a brief conversation on the official Arduino IRC with a lead developer determined that the libraries needed were no longer present in the IDE.

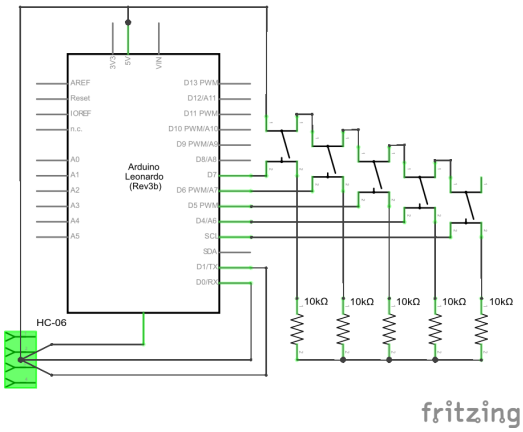

Figure 1- A circuit diagram of the complete system

After producing a working Media Library it was discovered that some of the HEX codes were outdated and were no longer supported by most media systems. This was a simple issue to solve, using the USB HID manual from earlier [6] allowed for quick look-up of the most update HEX codes and how they would be interpreted by the PC.

2.1 Practical and Experimental Work

With the second phase of research complete, the production of custom Media Key library was under taken and completed within the span of working week. This allowed for plenty of time to test, debug and fine tune the library against a number of media systems, including; Google Play Music, Windows Media Player, VLC, Winamp and Spotify.

With the successful completion of the required library, the first major Arduino Sprint has come to a close. As it stands at the time of writing, the system comprises of five push switches, each of which calls a different media control (namely previous, play, next, volume up and volume down). To ensure accurate readings from the switches, an external library was utilized. The Bounce2 library offers accurate edge detection and ensures that a function will only be called once per button press. While it could be said that utilizing an external library reduces the overall complexity of the system, the fact remains that main focus of the project is the media controls it offers. If time allows a bespoke debounce library may be produced but as it stands currently, the Bounce2 library provides the exact functionality required and allowed for more efficient testing.

2.3 Project Management

The previous report highlighted a list of milestones to be completed over the course of the project. The updated list can be seen below:

- ✓ Learn to correctly wire a microcontroller / breadboard device.
- ✓ Using a microcontroller as a basic human interface device. E.g. printing text in a document.
- ✓ Using a microcontroller to send media control signals to a PC. E.g. play/pause music.
- Add basic Bluetooth functionality.
- Develop an Android application to interface with a microcontroller.

A more detailed Gantt chart has been produced, to more display the time frame of the project accurately. The completed tasks are depicted in blue, while those yet to be started are in orange.

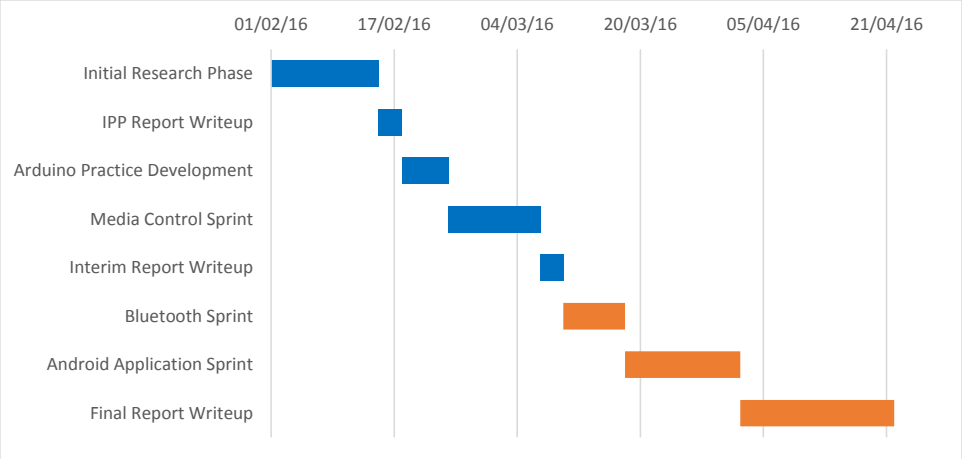

Figure 2 - A Gantt chart indicating the project timeline

As can be seen, the project is well underway and is on time to be complete for the deadline.

3 Future Programme of Work

3.1 Future Sprints

As one can see from the project timeline, two further sprints remain before completion of the project, namely the Bluetooth and Android Application sprints.

The goal of the next development sprint is to set up and configure basic Bluetooth communication with the Media controller. At the time of writing, the sprint has been delayed until the arrival of the

required hardware, namely a HC-06 Bluetooth Module. Once the appropriate resources are present, the scheduled sprint should be relatively short and the project will move into its final phase.

Once the Bluetooth module has arrived and been configured, the development of the companion Android application can begin. Mobile development can often be a time consuming process. Fortunately, the application to accompany the media control is of a minimalist design, in the hopes of keeping the operation and development as simple as possible. As seen in the timeline, the application sprint has been scheduled to be the longest. This is to ensure that application can be toughly tested, resulting in an enjoyable user experience.

3.2 Final Report

Once all the design, implementation and testing of the system is complete, all that remains of the project is the final report. At this time, the final report consists of rough notes and diagrams and the final section headers are yet to be finalized. While the report itself may currently be in a state of disarray, the notes being taken at stage of the project are thorough and should allow for a concise and balanced summary of the project.

4 Issues and Concerns

4.1 Hardware Concerns

As it stands, the only issue in terms of hardware is the lack of it. Once the required HC-06 module arrives, it should be a simple task to get it configured, thanks to prior experience with this technology.

4.2 Software Concerns

As previously stated, application development can often be a time consuming affair. The time scale for the project has left ample time to produce an application for the device. However, if more time is required to complete the final report, the overall quality of the application may suffer.

5 Conclusion

5.1 Summary

The project is well under way and is on track to be completed to a high standard for the deadline. The only constraint that currently exists is the time required for the Bluetooth module to arrive. Once Bluetooth implementation is complete, the only concern will be the time required to complete the Android application. While a very basic application can be created in a short amount of time, the usability and features may suffer if any major road-blocks are met during the sprint.

Compared to the time and effort required for the research and development of the project, the final report and presentation should prove to be a relatively simple affair. The hard work put in at the beginning of the project should allow for fast and efficient production of the final documentation.

6 References

6.1 References List

[1] “Arduino - ArduinoBoardLenoardo,” Arduino, [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardLeonardo. [Accessed 8 March 20016].

[2] “Arduino - ArduinoBoardUno,” Arduino, [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno. [Accessed 8 March 2016].

[3] M. Heironimus, “Add USB Game Controller to Arduino Leonardo/Micro,” instructables, [Online]. Available: http://www.instructables.com/id/Add-USB-Game-Controller-to-Arduino-LeonardoMicro/. [Accessed 8 March 2016].

[4] “arduino/Arduino: open-source electronics prototyping platform,” Arduino, [Online]. Available: https://github.com/arduino/arduino. [Accessed 8 March 2016].

[5] “USB.org - HID Tools,” USB Implementers Forum, [Online]. Available: http://www.usb.org/developers/hidpage/. [Accessed 8 March 2016].

[6] “Universal Serial Bus (USB) HID Usage Tables,” 21 October 2004. [Online]. Available: http://www.usb.org/developers/hidpage/Hut1_12v2.pdf. [Accessed 8 March 2016].

[7] S. Jones, “Arduino Leonardo Remote Control and Multimedia Keys Support in 1.0.5 | Some Thoughts,” [Online]. Available: http://stefanjones.ca/blog/arduino-leonardo-remote-multimedia-keys/. [Accessed 8 March 2016].