ESE 519 - Lab 0

Vaibhav N. Bhat, Shanjit S. Jajmann

February 8, 2015

Hardware connections

For this lab we have used the following hardware,

1. mbed (LPC1768)
2. Keypad
3. Buzzer
4. Breadboard and connecting wires
5. Seven Segment Display

The following is the schematic we used to interconnect the hardware using Fritzing,

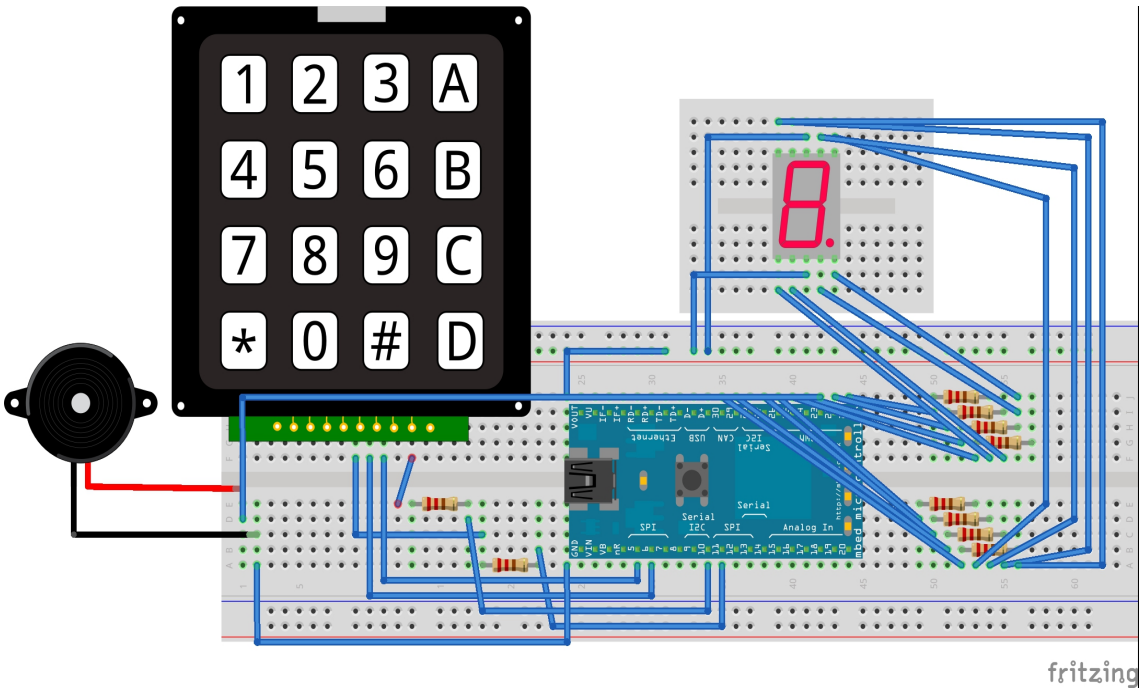

Figure 1: Hardware Connections

Part 0

In part0 of the assignment, we decided to use the value from the ADC to change the frequency of an LED blinky.

We implemented the following code,

```
/*
ese519-lab1-part0
    Vaibhav N. Bhat (vaibhavn@seas)
    Shanjit S. Jajmann (sjajmann@seas)

Idea : Vary the frequency of the led blink by reading
   randomness from the universe using onboard ADC

*/

#include "mbed.h"

// define leds
DigitalOut myled1(LED1);
DigitalOut myled2(LED2);
DigitalOut myled3(LED3);
DigitalOut myled4(LED4);

// analog pins to read randomness from the universe
AnalogIn ain(p19);
int a[4];

// serial interface - for debug
Serial pc(USBTX, USBRX);

// ticker timer for 18 second window
Ticker ticker_period_18;

// read the adc and assign digits to array 'a'
void get_adc()
{
    // This gets called every 18 seconds and updates the
       'a' array
    int temp;
    temp = ain.read_u16();
    a[0] = temp%10;
    temp = temp/10;
    a[1] = temp%10;
    temp = temp/10;
    a[2] = temp%10;
    temp = temp/10;
    a[3] = temp%10;

    // for debug - to see randomness on terminal
```

```
43      pc.printf("Bits are %d %d %d %d\n",a[0],a[1],a[2],a
            [3]);
44
45  }
46
47  int main() {
48
49    // serial speed
50    pc.baud(9600);
51
52
53    // 18 second timer period
54    ticker_period_18.attach(&get_adc,18.0);
55    get_adc();
56    while(1) {
57
58      // led1
59      myled1 = 1;
60      wait(0.1*a[0]);
61      myled1 = 0;
62
63      //led2
64      myled2 = 1;
65      wait(0.1*a[1]);
66      myled2 = 0;
67
68      //led3
69      myled3 = 1;
70      wait(0.1*a[2]);
71      myled3 = 0;
72
73      //led4
74      myled4 = 1;
75      wait(0.1*a[3]);
76      myled4 = 0;
77
78    }
79
80  }
```

Part 1

In part1 of the assignment, we have integrated the keypad with the mbed using polling. This involved checking in a while loop if a particular key of the keypad was pressed and light up corresponding leds.

When both the buttons are pressed simulataneously, any of the two leds can glow depending on which instruction the microcontroller is executing and that moment. The rate at which we are polling is the twice the amount of time to required to blink the leds and the associate wait.

Yes, it is possible to simulataneously type faster than a poll generally, in our case it depends on when the button is pressed. This is not the case when interrupts are used instead.

We implemented the following code,

```
/*
ese519-lab1-part0
    Vaibhav N. Bhat (vaibhavn@seas)
    Shanjit S. Jajmann (sjajmann@seas)

Idea : Vary the frequency of the led blink by reading
   randomness from the universe using onboard ADC

*/

#include "mbed.h"

// define leds
DigitalOut myled1(LED1);
DigitalOut myled2(LED2);
DigitalOut myled3(LED3);
DigitalOut myled4(LED4);

// analog pins to read randomness from the universe
AnalogIn ain(p19);
int a[4];

// serial interface - for debug
Serial pc(USBTX, USBRX);

// ticker timer for 18 second window
Ticker ticker_period_18;

// read the adc and assign digits to array 'a'
void get_adc()
{
    // This gets called every 18 seconds and updates the
       'a' array
    int temp;
    temp = ain.read_u16();
    a[0] = temp%10;
    temp = temp/10;
    a[1] = temp%10;
    temp = temp/10;
    a[2] = temp%10;
    temp = temp/10;
    a[3] = temp%10;

    // for debug - to see randomness on terminal
    pc.printf("Bits are %d %d %d %d\n",a[0],a[1],a[2],a
```

```
        [3]);
44
45  }
46
47  int main() {
48
49    // serial speed
50    pc.baud(9600);
51
52
53    // 18 second timer period
54    ticker_period_18.attach(&get_adc,18.0);
55    get_adc();
56    while(1) {
57
58      // led1
59      myled1 = 1;
60      wait(0.1*a[0]);
61      myled1 = 0;
62
63      //led2
64      myled2 = 1;
65      wait(0.1*a[1]);
66      myled2 = 0;
67
68      //led3
69      myled3 = 1;
70      wait(0.1*a[2]);
71      myled3 = 0;
72
73      //led4
74      myled4 = 1;
75      wait(0.1*a[3]);
76      myled4 = 0;
77
78    }
79
80  }
```

Part 2

In part2 of the assignment, instead of polling as in part1 we used interrupts on the keys to check if a key was pressed.

We implemented the following code,

```
1  /*
2  ese519-lab1-part2
3      Vaibhav N. Bhat (vaibhavn@seas)
4      Shanjit S. Jajmann (sjajmann@seas)
```

```
5
6  Idea : Using the keypad, interrupt when the key is
       pressed.
7
8  */
9
10 #include "mbed.h"
11
12
13 // Connections on the mbed
14 // Mbed - Keypad
15 // p5 - 7
16 // p6 - 6
17 // p7 - 5
18 // p8 - 3
19 // p10 - 8
20 // p11 - 4
21
22
23 //define the leds
24 DigitalOut myled1(LED1);
25 DigitalOut myled2(LED2);
26 DigitalOut myled3(LED3);
27 DigitalOut myled4(LED4);
28
29 //define the pins for # and 2
30 DigitalOut pin1(p10);
31 DigitalOut pin2(p11);
32
33
34 // keypad association
35 DigitalIn col4(p5);
36 InterruptIn col3(p6);
37 InterruptIn col2(p7);
38 DigitalIn col1(p8);
39
40 // ISR when button pressed - lights up led1
41 void light0(){
42         myled1 = 1;
43         wait(0.2);
44         myled1 = 0;
45         wait(0.2);
46     }
47
48 // ISR when button pressed - lights up led2
49 void light1(){
50         myled2 = 1;
51         wait(0.2);
52         myled2 = 0;
53         wait(0.2);
```

```
54 }
55
56 int main() {
57
58     pin1 = 1 ;
59     pin2 = 1 ;
60
61     // associate interrupts with functions light1 and
       light0
62     col3.rise(&light1);
63     col2.rise(&light0);
64     while(1) {
65     }
66 }
```

Part 3

In part3 of the assignment, we extended the part2 of the lab and used timer alonside interrupts to implement a dot and dash when the hash (#) key is pressed.

By using the dots and dash, we implemented the Morse Code. We implemented the following code,

```
1  /*
2  ese519-lab1-part3
3      Vaibhav N. Bhat (vaibhavn@seas)
4      Shanjit S. Jajmann (sjajmann@seas)
5
6  Idea : Using the keypad and timers, implement dot and
       dash functionality
7  Timer 1 is used to measure the duration of the key press
8  Timer 2 is used independently to generate spaces
9
10 */
11
12 #include "mbed.h"
13
14
15 // Connections on the mbed
16 // Mbed - Keypad
17 // p5 - 7
18 // p6 - 6
19 // p7 - 5
20 // p8 - 3
21 // p10 - 8
22 // p11 - 4
23
24 bool flag = false;
25
```

```
26 // Interrupt whenever pin is pressed (#) - keypad
27 InterruptIn key(p6);
28 DigitalOut row(p11);
29
30 // led interface - visual feedback
31 DigitalOut myled1(LED1);
32 DigitalOut myled2(LED2);
33 DigitalOut myled3(LED3);
34 DigitalOut myled4(LED4);
35
36 /*
37 *   Dot represented by led1
38 *   Dash represented by led2
39 *   Space represented by led3
40 */
41
42 // serial interface - for debugging
43 Serial pc(USBTX, USBRX);
44
45 // timers to ensure dot and dash
46 Timer timer_400;
47 Timer timer_key;
48
49 // ISR when the key is just pressed on the keypad (for
       debugging)
50 void key_rise_int1()
51 {
52     // debouncing code - wait and check
53     wait(0.01);
54     if(key)
55     {
56         // serial debug
57         pc.printf("pressed");
58     }
59
60 }
61
62 // ISR when the key is just pressed on the keypad (for
       debugging)
63 void key_fall_int1()
64 {
65     // debouncing code - wait and check
66     wait(0.01);
67     if(!key)
68     {
69         // serial debug
70         pc.printf("released");
71
72     }
73 }
```

```
74
75
76  // ISR when the key is just pressed on the keypad
77  void key_rise_int()
78  {
79      // debouncing code - wait and check
80      //read the value of the key after 10ms and set the
            flag
81      wait(0.01);
82
83      if(key)
84      {
85          timer_400.stop();    // Stop the space timer
86          flag = true;
87          //start a timer
88          timer_key.start();  // Start counting the
                duration of the key press
89          pc.printf("key pressed");    // For debugging
90      }
91
92      else
93      {
94          flag = false;
95      }
96
97  }
98
99  // ISR when the key is just pressed on the keypad
100 void key_fall_int()
101 {
102     wait(0.01);
103
104     if(!key)
105     {
106         pc.printf("key released");  // For debugging
107         //timer_400.reset();
108         // stop the timer
109         timer_key.stop();    // Stop the first timer which
                measures the key press
110
111         // read the timer value and decide if dot, dash
112         int timer_key_val = timer_key.read_ms();
113
114         if((timer_key_val>40)&&(timer_key_val<220)&&(flag
                ==true) )  // Dot has been pressed
115         {
116             myled1 = 1;
117             wait(0.05);
118             myled1 = 0;
119         }
```

```
120
121         else if (timer_key_val>220){     // Dash has been
                pressed
122             myled2 = 1;
123             wait(0.05);
124             myled2 = 0;
125         }
126
127         // Reset and stop the measurement timer
128         timer_key.stop();
129         timer_key.reset();
130         flag=0;
131         timer_400.reset();  // Reset and restart the
                timer for space
132         timer_400.start();
133
134     }
135 }
136
137
138 int main() {
139
140     myled1 = 0;      // blink the leds once to indicate
            start of the program
141     myled2 = 1;
142     myled3 = 1;
143     myled4 = 1;
144
145     wait(0.5);
146     myled1 = 0;
147     myled2 = 0;
148     myled3 = 0;
149     myled4 = 0;
150
151     pc.baud(9600);
152     row = 1;
153
154
155     // start the timer for 400ms to check the occurence
            of space
156     timer_400.start();
157
158     key.rise(&key_rise_int);     // Set up the button
            interrupt handlers
159     key.fall(&key_fall_int);
160
161
162     while(1) {
163         // if timer value hits 400 ms, then reset the
                timer and read as space.
```

```
164             if(timer_400.read_ms()>=400)
165             {
166             timer_400.reset();
167             if(!flag)
168             {
169             //pc.printf("Got a space\n");
170             myled3 = 1;      //Indicate the presence of space
                    by blinking led
171             wait(0.05);
172             myled3 = 0;
173             }
174             }
175         }
176 }
```

Part 4

In part4 of the assignment, we again extended part3 of the lab and integrated a common anode based seven segment display with our existing hardware connections for part3.

We integrated this with our existing Morse Code hardware in part3, to show different patterns on the seven segment.

We implemented the seven segment by individually switching on/off the required leds.

Code:

```
1  #include "mbed.h"
2  /*
3  ese519-lab1-part4
4      Vaibhav N. Bhat (vaibhavn@seas)
5      Shanjit S. Jajmann (sjajmann@seas)
6
7  Idea : Using the keypad and timers, implement dot and
       dash functionality
8  Timer 1 is used to measure the duration of the key press
9  Timer 2 is used independently to generate spaces
10
11 7 segment display is interfaced to display dot, space and
       dash
12
13 Dot - Segment D
14 Dash - Segment G
15 Space - Segment A
16 */
17
18 // Connections on the mbed
19 // Mbed - Keypad
20 // p5 - 7
21 // p6 - 6
```

```
22 // p7 - 5
23 // p8 - 3
24 // p10 - 8
25 // p11 - 4
26
27
28 bool flag = false;
29
30 // Define the row and column for keypad
31 InterruptIn key(p6);
32 DigitalOut row(p11);
33
34 DigitalOut myled1(LED1);
35 DigitalOut myled2(LED2);
36 DigitalOut myled3(LED3);
37 DigitalOut myled4(LED4);
38 Serial pc(USBTX, USBRX); // tx, rx
39
40 // 2 timers used for space and dot/dash measurement
41 Timer timer_400;
42 Timer timer_key;
43
44 // seven segment display interface
45 // Mbed - Seven Segment (common anode)
46 // p21 - 1
47 // p22 - 2
48 // gnd - 3
49 // p24 - 4
50 // p25 - 5
51 // p26 - 6
52 // p27 - 7
53 // gnd - 8
54 // p29 - 9
55 // p30 - 10
56
57 DigitalOut ss_a(p27);
58 DigitalOut ss_b(p26);
59 DigitalOut ss_c(p24);
60 DigitalOut ss_d(p22);
61 DigitalOut ss_e(p21);
62 DigitalOut ss_f(p29);
63 DigitalOut ss_g(p30);
64 DigitalOut ss_dot(p25);
65
66 // Function for handling dot behavior
67 inline void dot()
68 {
69 ss_d = 0;
70 wait(0.05);
71 ss_d = 1;
```

```
72
73  }
74
75  // Function for handling dash behavior
76  inline void dash()
77  {
78      ss_g = 0;
79      wait(0.05);
80      ss_g = 1;
81
82  }
83
84  // Function for handling space behavior
85  inline void space()
86  {
87      ss_a = 0;
88      wait(0.05);
89      ss_a = 1;
90
91  }
92  // Function for turning of the 7 segment display
93
94  void segment_off()
95  {
96      ss_a = 1;
97      ss_b = 1;
98      ss_c = 1;
99      ss_d = 1;
100     ss_e = 1;
101     ss_f = 1;
102     ss_g = 1;
103     ss_dot = 1;
104
105 }
106
107 // Function for turning on all leds in the 7 segment
        display
108 // Used for debugging
109 void segment_on()
110 {
111     ss_a = 0;
112     ss_b = 0;
113     ss_c = 0;
114     ss_d = 0;
115     ss_e = 0;
116     ss_f = 0;
117     ss_g = 0;
118     ss_dot = 0;
119 }
120
```

```
121 // Interrupt handler for key press - Debugging purpose
122 void key_rise_int1()
123 {
124     wait(0.01); // Debounce period
125     if(key)
126     {
127         pc.printf("pressed");
128     }
129
130 }
131
132 // Interrupt handler for key release - Debugging purpose
133 void key_fall_int1()
134 {
135     wait(0.01);
136     if(!key)
137     {
138         pc.printf("released");
139
140     }
141 }
142
143 // Interrupt handler for key press
144 void key_rise_int()
145 {
146
147     wait(0.01);
148     //read the value of the key after 10ms and set the
        flag
149     if(key)
150     {
151         timer_400.stop();   // Stop the space timer
152         flag = true;
153         //start a timer for measuring the duration of the
            key press
154         timer_key.start();
155         pc.printf("key pressed");
156     }
157
158     else
159     {
160         flag = false;    // Debouncing failed- key wasn't
            pressed
161     }
162
163 }
164
165 // Interrupt handler for key release
166 void key_fall_int()
167 {
```

```
168         wait(0.01);
169
170         if(!key)
171         {
172             pc.printf("key released");
173             //timer_400.reset();
174             // stop the timer used for measuring key press
175             timer_key.stop();
176
177             // read the timer value and decide if dot, dash
178             int timer_key_val = timer_key.read_ms();
179
180             if((timer_key_val>40)&&(timer_key_val<220)&&(flag
                    ==true) )
181             {
182                 /*myled1 = 1;
183                 wait(0.05);
184                 myled1 = 0;*/
185                 dot();  // Dot pressed
186             }
187
188             else if (timer_key_val>220){
189                 /*myled2 = 1;
190                 wait(0.05);
191                 myled2 = 0;*/
192                 dash(); //Dash pressed
193             }
194
195             // reset the measurement timer
196             timer_key.stop();
197             timer_key.reset();
198             flag=0;
199             timer_400.reset();  // Restart the space timer
                    after reset
200             timer_400.start();
201
202         }
203     }
204
205
206     int main() {
207
208
209         // Check the leds
210         myled1 = 0;
211         myled2 = 1;
212         myled3 = 1;
213         myled4 = 1;
214
215         wait(0.5);
```

```
216         myled1 = 0;
217         myled2 = 0;
218         myled3 = 0;
219         myled4 = 0;
220 
221         // Check the seven segment
222         segment_off();
223         wait(0.5);
224         segment_on();
225         wait(0.5);
226         segment_off();
227 
228         pc.baud(9600);
229         row = 1;
230 
231 
232         // start the timer for 400ms
233         timer_400.start();
234 
235         key.rise(&key_rise_int);
236         key.fall(&key_fall_int);     // Set up the interrupt
                handlers
237 
238 
239         while(1) {
240             // if timer value hits 400 ms, then reset the
                    timer and read as space.
241             if(timer_400.read_ms()>=400)
242             {
243             timer_400.reset();
244 
245             if(!flag)
246             {
247             //pc.printf("Got a space\n");
248             /*myled3 = 1;
249             wait(0.05);
250             myled3 = 0;*/
251             space();     // Space occured
252             }
253             }
254         }
255 }
```

Part 5

In part5 of the assignment, we used the seven segment to display ASCII digits on the seven segment, we implemented this by hard coding in what leds need to glow for what characters.

With this integrated, we were able to show different ASCII characters for corresponding dots and dashes on the seven segment display.

Code:

```
#include "mbed.h"

#define DOT 1
#define DASH 2
/*
ese519-lab1-part5
    Vaibhav N. Bhat (vaibhavn@seas)
    Shanjit S. Jajmann (sjajmann@seas)

Idea : Using the keypad and timers, implement dot and dash functionality
Timer 1 is used to measure the duration of the key press
Timer 2 is used independently to generate spaces

7 segment display is interfaced to display dot, space and dash
The ASCII characters are interpreted and printed on the serial terminal

*/

// Connections on the mbed
// Mbed - Keypad
// p5 - 7
// p6 - 6
// p7 - 5
// p8 - 3
// p10 - 8
// p11 - 4

// array for saving the received dot / dash pattern
int ch[5];
int count_ch = 0;   // stores count of the number of dots/dashes received

// Flag for detecting key
bool flag = false;

// Set up the row and column for the keypad
// p6 senses active high and p11 activates the row
InterruptIn key(p6);
DigitalOut row(p11);

// leds
DigitalOut myled1(LED1);
DigitalOut myled2(LED2);
```

```
43 DigitalOut myled3(LED3);
44 DigitalOut myled4(LED4);
45 Serial pc(USBTX, USBRX); // tx, rx
46
47 // Timers used for detecting dot/dash and space
48 Timer timer_400;     // Timer for space
49 Timer timer_key;
50
51 // seven segment display
52 // Mbed - Seven Segment (common anode)
53 // p21 - 1
54 // p22 - 2
55 // gnd - 3
56 // p24 - 4
57 // p25 - 5
58 // p26 - 6
59 // p27 - 7
60 // gnd - 8
61 // p29 - 9
62 // p30 - 10
63 DigitalOut ss_a(p27);
64 DigitalOut ss_b(p26);
65 DigitalOut ss_c(p24);
66 DigitalOut ss_d(p22);
67 DigitalOut ss_e(p21);
68 DigitalOut ss_f(p29);
69 DigitalOut ss_g(p30);
70 DigitalOut ss_dot(p25);
71
72 // Function for determing the character after storing the
       dot/dash pattern received
73 inline void find_char()
74 {
75
76     if(count_ch==0)
77     {
78         return;
79     }
80
81     else if(count_ch==1)     // If the no of characters
          received is 1
82     {                        // the letter can be either E
          or T
83         if(ch[0]==DOT)
84         {
85         pc.printf("E");     // Print the interpreted
              character on the terminal
86         }
87
88         else if (ch[0]==DASH)
```

```
89              {
90              pc.printf("T");
91              }
92          }
93
94          else if(count_ch==2)
95          {
96              if((ch[0]==DOT)&&(ch[1]==DASH))
97              {
98                  pc.printf("A");
99              }
100
101             else if((ch[0]==DOT)&&(ch[1]==DOT))
102             {
103                 pc.printf("I");
104             }
105
106             else if((ch[0]==DASH)&&(ch[1]==DOT))
107             {
108                 pc.printf("N");
109             }
110
111             else if((ch[0]==DASH)&&(ch[1]==DASH))
112             {
113                 pc.printf("M");
114             }
115         }
116
117         else if(count_ch==3)
118         {
119             if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT))
120             {
121                 pc.printf("S");
122             }
123
124             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT)
                    )
125             {
126                 pc.printf("R");
127             }
128
129             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
                    ))
130             {
131                 pc.printf("W");
132             }
133
134             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    )
135             {
```

```
136                 pc.printf("D");
137             }
138
139             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
                    ))
140             {
141                 pc.printf("G");
142             }
143
144             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
                    ))
145             {
146                 pc.printf("K");
147             }
148
149             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
                    DASH))
150             {
151                 pc.printf("O");
152             }
153
154             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
                    )
155             {
156                 pc.printf("U");
157             }
158
159
160         }
161
162         else if(count_ch==4)
163         {
164             if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch
                    [3]==DOT))
165             {
166                 pc.printf("H");
167             }
168
169             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT)
                    &&(ch[3]==DOT))
170             {
171                 pc.printf("L");
172             }
173
174             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
                    &&(ch[3]==DOT))
175             {
176                 pc.printf("F");
177             }
178
```

```
179             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DASH))
180             {
181                 pc.printf("V");
182             }
183
184             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
                    )&&(ch[3]==DOT))
185             {
186                 pc.printf("Z");
187             }
188
189             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
                    )&&(ch[3]==DOT))
190             {
191                 pc.printf("P");
192             }
193
194             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
                    )&&(ch[3]==DASH))
195             {
196                 pc.printf("J");
197             }
198
199             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
                    )&&(ch[3]==DASH))
200             {
201                 pc.printf("Y");
202             }
203
204             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DASH))
205             {
206                 pc.printf("X");
207             }
208
209             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
                    )&&(ch[3]==DASH))
210             {
211                 pc.printf("Q");
212             }
213
214             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
                    )&&(ch[3]==DOT))
215             {
216                 pc.printf("C");
217             }
218
219             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DOT))
```

```
220             {
221                 pc.printf("B");
222             }
223         }
224
225         else if(count_ch==5)
226         {
227             if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch
                    [3]==DOT)&&(ch[4]==DOT))
228             {
229                 pc.printf("5");
230             }
231
232             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DOT)&&(ch[4]==DASH))
233             {
234                 pc.printf("4");
235             }
236
237             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DASH)&&(ch[4]==DASH))
238             {
239                 pc.printf("3");
240             }
241
242             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
                    &&(ch[3]==DASH)&&(ch[4]==DASH))
243             {
244                 pc.printf("2");
245             }
246
247             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
                    )&&(ch[3]==DASH)&&(ch[4]==DASH))
248             {
249                 pc.printf("1");
250             }
251
252             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DOT)&&(ch[4]==DOT))
253             {
254                 pc.printf("6");
255             }
256
257             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
                    )&&(ch[3]==DOT)&&(ch[4]==DOT))
258             {
259                 pc.printf("7");
260             }
261
262             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
```

```
                DASH)&&(ch[3]==DOT)&&(ch[4]==DOT))
263         {
264             pc.printf("8");
265         }
266
267         else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
                DASH)&&(ch[3]==DASH)&&(ch[4]==DOT))
268         {
269             pc.printf("9");
270         }
271
272         else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
                DASH)&&(ch[3]==DASH)&&(ch[4]==DASH))
273         {
274             pc.printf("0");
275         }
276
277     }
278
279
280 }
281
282 // Function for space behavior
283 inline void space()
284 {
285     ss_a = 0;
286     wait(0.05);
287     ss_a = 1;
288
289     // depending on count, determine the character
290     find_char();
291
292     count_ch = 0;    // Reset the number of characters
            count
293     ch[0] = 0;       // Clear the array
294     ch[1] = 0;
295     ch[2] = 0;
296     ch[3] = 0;
297     ch[4] = 0;
298
299 }
300
301 inline void dot()
302 {
303     ss_d = 0;
304     wait(0.05);
305     ss_d = 1;
306     if(count_ch>=5)
307     {
308     space();    // When count >= 5 the array must be
```

```
            checked for the ascii character
309     }
310     else
311     {
312         ch[count_ch] = DOT;
313         count_ch++;
314     }
315 }
316
317 inline void dash()
318 {
319     ss_g = 0;
320     wait(0.05);
321     ss_g = 1;
322
323     if(count_ch>=5) // When count >= 5 the array must be
            checked for the ascii character
324     {
325     space();
326     }
327     else
328     {
329         ch[count_ch] = DASH;
330         count_ch++;
331     }}
332
333 // function to turn off the 7 segment
334 void segment_off()
335 {
336     ss_a = 1;
337     ss_b = 1;
338     ss_c = 1;
339     ss_d = 1;
340     ss_e = 1;
341     ss_f = 1;
342     ss_g = 1;
343     ss_dot = 1;
344
345 }
346 // function to turn on the 7 segment
347 void segment_on()
348 {
349     ss_a = 0;
350     ss_b = 0;
351     ss_c = 0;
352     ss_d = 0;
353     ss_e = 0;
354     ss_f = 0;
355     ss_g = 0;
356     ss_dot = 0;
```

```
357 }
358
359 // debugging function
360 void key_rise_int1()
361 {
362     wait(0.01);
363     if(key)
364     {
365         pc.printf("pressed");
366     }
367
368 }
369
370 // debugging function
371 void key_fall_int1()
372 {
373     wait(0.01);
374     if(!key)
375     {
376         pc.printf("released");
377
378     }
379 }
380
381 // Interrupt handler for key press
382 void key_rise_int()
383 {
384
385     wait(0.01);
386     //read the value of the key after 10ms and set the
        flag
387     if(key)
388     {
389         timer_400.stop();
390         flag = true;
391         //start a timer
392         timer_key.start();
393        // pc.printf("key pressed");
394     }
395
396     else
397     {
398         flag = false;
399     }
400
401 }
402 // Interrupt handler for key press
403 void key_fall_int()
404 {
405     wait(0.01);
```

```
406
407         if(!key)
408         {
409             //pc.printf("key released");
410             //timer_400.reset();
411             // stop the timer
412             timer_key.stop();
413
414             // read the timer value and decide if dot, dash
415             int timer_key_val = timer_key.read_ms();
416
417             if((timer_key_val>40)&&(timer_key_val<220)&&(flag
                    ==true) )
418             {
419                 /*myled1 = 1;
420                 wait(0.05);
421                 myled1 = 0;*/
422                 dot();
423             }
424
425             else if (timer_key_val>220){
426                 /*myled2 = 1;
427                 wait(0.05);
428                 myled2 = 0;*/
429                 dash();
430             }
431
432             // reset the timer
433             timer_key.stop();    // Stop and reset the dot/
                    dash timer
434             timer_key.reset();
435             flag=0;
436             timer_400.reset(); // Reset the space timer
437             timer_400.start();
438
439         }
440     }
441
442
443     int main() {
444
445
446         // Check the leds
447         myled1 = 0;
448         myled2 = 1;
449         myled3 = 1;
450         myled4 = 1;
451
452         wait(0.5);
453         myled1 = 0;
```

```
454         myled2 = 0;
455         myled3 = 0;
456         myled4 = 0;
457 
458         // Check the seven segment
459         segment_off();
460         wait(0.5);
461         segment_on();
462         wait(0.5);
463         segment_off();
464 
465         pc.baud(9600);
466         row = 1;
467 
468 
469         // start the timer for 400ms
470         timer_400.start();
471 
472         key.rise(&key_rise_int);      // Setup the interrupt
                handlers
473         key.fall(&key_fall_int);
474 
475 
476         while(1) {
477             // if timer value hits 400 ms, then reset the
                    timer and read as space.
478             if(timer_400.read_ms()>=400)
479             {
480             timer_400.reset();
481 
482             if(!flag)
483             {
484             //pc.printf("Got a space\n");
485             /*myled3 = 1;
486             wait(0.05);
487             myled3 = 0;*/
488             space();
489             }
490             }
491         }
492 }
```

Extra Credit 1

For this extra credit, we integrated the buzzer on a particular pin of the mbed and changed the frequency for which it beeps to distinguish between a dot and a dash.

We used the following code,

Code:

```
1  #include "mbed.h"
2
3  #define DOT 1
4  #define DASH 2
5  /*
6  ese519-lab1-Extra Credit Part 1
7       Vaibhav N. Bhat (vaibhavn@seas)
8       Shanjit S. Jajmann (sjajmann@seas)
9
10 Idea : Using the keypad and timers, implement dot and
       dash functionality
11 Timer 1 is used to measure the duration of the key press
12 Timer 2 is used independently to generate spaces
13
14 7 segment display is interfaced to display dot, space and
       dash
15 The ASCII characters are interpreted and printed on the
       serial terminal
16 Buzzer is interfaced to beep when a dot is pressed and a
       longer beep
17 when a dash is pressed
18
19 */
20
21 // Connections on the mbed
22 // Mbed - Keypad
23 // p5 - 7
24 // p6 - 6
25 // p7 - 5
26 // p8 - 3
27 // p10 - 8
28 // p11 - 4
29
30 // array for saving the received dot / dash pattern
31 int ch[5];
32 int count_ch = 0;    // stores count of the number of dots
       /dashes received
33
34 // Flag for detecting key
35 bool flag = false;
36
37 // Set up the row and column for the keypad
38 // p6 senses active high and p11 activates the row
39 InterruptIn key(p6);
40 DigitalOut row(p11);
41
42 // leds
43 DigitalOut myled1(LED1);
44 DigitalOut myled2(LED2);
45 DigitalOut myled3(LED3);
```

```
46 DigitalOut myled4(LED4);
47 Serial pc(USBTX, USBRX); // tx, rx
48
49 // Timers used for detecting dot/dash and space
50 Timer timer_400;    // Timer for space
51 Timer timer_key;
52
53 // seven segment display
54 // Mbed - Seven Segment (common anode)
55 // p21 - 1
56 // p22 - 2
57 // gnd - 3
58 // p24 - 4
59 // p25 - 5
60 // p26 - 6
61 // p27 - 7
62 // gnd - 8
63 // p29 - 9
64 // p30 - 10
65 DigitalOut ss_a(p27);
66 DigitalOut ss_b(p26);
67 DigitalOut ss_c(p24);
68 DigitalOut ss_d(p22);
69 DigitalOut ss_e(p21);
70 DigitalOut ss_f(p29);
71 DigitalOut ss_g(p30);
72 DigitalOut ss_dot(p25);
73
74 //buzzer
75 DigitalOut buzz(p23);
76
77 // Function for determing the character after storing the
       dot/dash pattern received
78 inline void find_char()
79 {
80
81     if(count_ch==0)
82     {
83         return;
84     }
85
86     else if(count_ch==1)    // If the no of characters
          received is 1
87     {                       // the letter can be either E
          or T
88         if(ch[0]==DOT)
89         {
90         pc.printf("E");     // Print the interpreted
              character on the terminal
91         }
```

```
92
93              else if (ch[0]==DASH)
94              {
95              pc.printf("T");
96              }
97          }
98
99          else if(count_ch==2)
100         {
101             if((ch[0]==DOT)&&(ch[1]==DASH))
102             {
103                 pc.printf("A");
104             }
105
106             else if((ch[0]==DOT)&&(ch[1]==DOT))
107             {
108                 pc.printf("I");
109             }
110
111             else if((ch[0]==DASH)&&(ch[1]==DOT))
112             {
113                 pc.printf("N");
114             }
115
116             else if((ch[0]==DASH)&&(ch[1]==DASH))
117             {
118                 pc.printf("M");
119             }
120         }
121
122         else if(count_ch==3)
123         {
124             if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT))
125             {
126                 pc.printf("S");
127             }
128
129             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT)
                    )
130             {
131                 pc.printf("R");
132             }
133
134             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
                    ))
135             {
136                 pc.printf("W");
137             }
138
139             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
```

```
            )
140         {
141             pc.printf("D");
142         }
143
144         else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
                ))
145         {
146             pc.printf("G");
147         }
148
149         else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
                ))
150         {
151             pc.printf("K");
152         }
153
154         else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
                DASH))
155         {
156             pc.printf("O");
157         }
158
159         else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
                )
160         {
161             pc.printf("U");
162         }
163
164
165     }
166
167     else if(count_ch==4)
168     {
169         if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch
                [3]==DOT))
170         {
171             pc.printf("H");
172         }
173
174         else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT)
                &&(ch[3]==DOT))
175         {
176             pc.printf("L");
177         }
178
179         else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
                &&(ch[3]==DOT))
180         {
181             pc.printf("F");
```

```
182             }
183
184             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DASH))
185             {
186                 pc.printf("V");
187             }
188
189             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
                    )&&(ch[3]==DOT))
190             {
191                 pc.printf("Z");
192             }
193
194             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
                    )&&(ch[3]==DOT))
195             {
196                 pc.printf("P");
197             }
198
199             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
                    )&&(ch[3]==DASH))
200             {
201                 pc.printf("J");
202             }
203
204             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
                    )&&(ch[3]==DASH))
205             {
206                 pc.printf("Y");
207             }
208
209             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DASH))
210             {
211                 pc.printf("X");
212             }
213
214             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
                    )&&(ch[3]==DASH))
215             {
216                 pc.printf("Q");
217             }
218
219             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
                    )&&(ch[3]==DOT))
220             {
221                 pc.printf("C");
222             }
223
```

```
224             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DOT))
225             {
226                 pc.printf("B");
227             }
228         }
229
230         else if(count_ch==5)
231         {
232             if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch
                    [3]==DOT)&&(ch[4]==DOT))
233             {
234                 pc.printf("5");
235             }
236
237             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DOT)&&(ch[4]==DASH))
238             {
239                 pc.printf("4");
240             }
241
242             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DASH)&&(ch[4]==DASH))
243             {
244                 pc.printf("3");
245             }
246
247             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
                    &&(ch[3]==DASH)&&(ch[4]==DASH))
248             {
249                 pc.printf("2");
250             }
251
252             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
                    )&&(ch[3]==DASH)&&(ch[4]==DASH))
253             {
254                 pc.printf("1");
255             }
256
257             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DOT)&&(ch[4]==DOT))
258             {
259                 pc.printf("6");
260             }
261
262             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
                    )&&(ch[3]==DOT)&&(ch[4]==DOT))
263             {
264                 pc.printf("7");
265             }
```

```
266
267             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
                    DASH)&&(ch[3]==DOT)&&(ch[4]==DOT))
268             {
269                 pc.printf("8");
270             }
271
272             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
                    DASH)&&(ch[3]==DASH)&&(ch[4]==DOT))
273             {
274                 pc.printf("9");
275             }
276
277             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
                    DASH)&&(ch[3]==DASH)&&(ch[4]==DASH))
278             {
279                 pc.printf("0");
280             }
281
282         }
283
284
285 }
286
287
288 inline void space()
289 {
290     ss_a = 0;
291     wait(0.05);
292     ss_a = 1;
293
294
295     // depending on count, determine the character
296     find_char();
297
298     count_ch = 0;
299     ch[0] = 0;
300     ch[1] = 0;
301     ch[2] = 0;
302     ch[3] = 0;
303     ch[4] = 0;
304
305 }
306
307 inline void dot()
308 {
309     ss_d = 0;
310     wait(0.05);
311     ss_d = 1;
312
```

```
313
314         buzz = 1;
315         wait(0.05);
316         buzz = 0;
317
318         if(count_ch>=5)
319         {
320         space();    // When count >= 5 the array must be
                checked for the ascii character
321         }
322         else
323         {
324             ch[count_ch] = DOT;
325             count_ch++;
326         }
327     }
328
329     inline void dash()
330     {
331         ss_g = 0;
332         wait(0.05);
333         ss_g = 1;
334
335         buzz = 1;
336         wait(0.15);
337         buzz = 0;
338
339         if(count_ch>=5)  // When count >= 5 the array must be
                checked for the ascii character
340         {
341         space();
342         }
343         else
344         {
345             ch[count_ch] = DASH;
346             count_ch++;
347         }}
348
349     // function to turn off the 7 segment
350     void segment_off()
351     {
352         ss_a = 1;
353         ss_b = 1;
354         ss_c = 1;
355         ss_d = 1;
356         ss_e = 1;
357         ss_f = 1;
358         ss_g = 1;
359         ss_dot = 1;
360
```

```
361 }
362
363 // function to turn on the 7 segment
364 void segment_on()
365 {
366     ss_a = 0;
367     ss_b = 0;
368     ss_c = 0;
369     ss_d = 0;
370     ss_e = 0;
371     ss_f = 0;
372     ss_g = 0;
373     ss_dot = 0;
374 }
375
376 // debugging function
377 void key_rise_int1()
378 {
379     wait(0.01);
380     if(key)
381     {
382         pc.printf("pressed");
383     }
384
385 }
386
387 // debugging function
388 void key_fall_int1()
389 {
390     wait(0.01);
391     if(!key)
392     {
393         pc.printf("released");
394
395     }
396 }
397
398 // Interrupt handler for key press
399 void key_rise_int()
400 {
401
402     wait(0.01);
403     //read the value of the key after 10ms and set the
        flag
404     if(key)
405     {
406         timer_400.stop();
407         flag = true;
408         //start a timer
409         timer_key.start();
```

```
410         // pc.printf("key pressed");
411     }
412 
413     else
414     {
415         flag = false;
416     }
417 
418 }
419 // Interrupt handler for key release
420 void key_fall_int()
421 {
422     wait(0.01);
423 
424     if(!key)
425     {
426         //pc.printf("key released");
427         //timer_400.reset();
428         // stop the timer
429         timer_key.stop();
430 
431         // read the timer value and decide if dot, dash
432         int timer_key_val = timer_key.read_ms();
433 
434         if((timer_key_val>40)&&(timer_key_val<220)&&(flag
                ==true) )
435         {
436             /*myled1 = 1;
437             wait(0.05);
438             myled1 = 0;*/
439             dot();
440         }
441 
442         else if (timer_key_val>220){
443             /*myled2 = 1;
444             wait(0.05);
445             myled2 = 0;*/
446             dash();
447         }
448 
449         // reset the timer
450         timer_key.stop();
451         timer_key.reset();
452         flag=0;
453         timer_400.reset();  // Reset the space timer
454         timer_400.start();
455 
456     }
457 }
458 
```

```
459
460 int main() {
461
462
463     // Check the leds
464     myled1 = 0;
465     myled2 = 1;
466     myled3 = 1;
467     myled4 = 1;
468
469     wait(0.5);
470     myled1 = 0;
471     myled2 = 0;
472     myled3 = 0;
473     myled4 = 0;
474
475     // Check the seven segment
476     segment_off();
477     wait(0.5);
478     segment_on();
479     wait(0.5);
480     segment_off();
481
482     pc.baud(9600);
483     row = 1;
484
485
486     // start the timer for 400ms
487     timer_400.start();
488
489     key.rise(&key_rise_int); // Setup the interrupt
            handlers
490     key.fall(&key_fall_int);
491
492
493     while(1) {
494         // if timer value hits 400 ms, then reset the
                timer and read as space.
495         if(timer_400.read_ms()>=400)
496         {
497         timer_400.reset();
498
499         if(!flag)
500         {
501         //pc.printf("Got a space\n");
502
503         space();
504         }
505         }
506     }
```

```
507 }
```

Extra Credit 2

For this extra credit, we implemented the space functionality to distinguish between two words when typed in the morse code. 5 or more spaces are intepreted as a single space, while a 2-3 spaces can be allowed when pressing the key. This configuration can be changed depending on the final end user.

Code:

```
1  #include "mbed.h"
2
3  #define DOT 1
4  #define DASH 2
5  /*
6  ese519-lab1-Extra Credit Seven segment
7       Vaibhav N. Bhat (vaibhavn@seas)
8       Shanjit S. Jajmann (sjajmann@seas)
9
10 Idea : Using the keypad and timers, implement dot and
       dash functionality
11 Timer 1 is used to measure the duration of the key press
12 Timer 2 is used independently to generate spaces
13
14 7 segment display is interfaced to display dot, space and
        dash
15 The ASCII characters are interpreted and printed on the
       serial terminal
16
17 Buzzer is interfaced to beep when a dot is pressed and a
       longer beep
18 when a dash is pressed
19
20 ASCII characters are display on the 7 segment display
21
22 2 spaces are allowed between letters.
23 5 or more spaces is considered as one space
24
25 */
26
27 // Connections on the mbed
28 // Mbed - Keypad
29 // p5 - 7
30 // p6 - 6
31 // p7 - 5
32 // p8 - 3
33 // p10 - 8
34 // p11 - 4
35
```

```
36 // array for saving the received dot / dash pattern
37 int ch[5];
38 int count_ch = 0;   // stores count of the number of dots
      /dashes received
39
40 bool flag = false;
41
42 // Set up the row and column for the keypad
43 // p6 senses active high and p11 activates the row
44 InterruptIn key(p6);
45 DigitalOut row(p11);
46
47 DigitalOut myled1(LED1);
48 DigitalOut myled2(LED2);
49 DigitalOut myled3(LED3);
50 DigitalOut myled4(LED4);
51 Serial pc(USBTX, USBRX); // tx, rx
52
53 // Timers used for detecting dot/dash and space
54 Timer timer_400;    // Timer for space
55 Timer timer_key;
56
57 DigitalOut buzz(p23);
58
59 // seven segment display
60 // Mbed - Seven Segment (common cathode)
61 // p21 - 1
62 // p22 - 2
63 // gnd - 3
64 // p24 - 4
65 // p25 - 5
66 // p26 - 6
67 // p27 - 7
68 // gnd - 8
69 // p29 - 9
70 // p30 - 10
71 DigitalOut ss_a(p27);
72 DigitalOut ss_b(p26);
73 DigitalOut ss_c(p24);
74 DigitalOut ss_d(p22);
75 DigitalOut ss_e(p21);
76 DigitalOut ss_f(p29);
77 DigitalOut ss_g(p30);
78 DigitalOut ss_dot(p25);
79
80 // Function which displays ASCII character on the 7
      segment display given a character
81 void seven_seg_disp(char sev_ch)
82 {
83     if(sev_ch == '0'){
```

```
84              ss_a = 0;
85              ss_b = 0;
86              ss_c = 0;
87              ss_d = 0;
88              ss_e = 0;
89              ss_f = 0;
90              ss_g = 1;
91              ss_dot = 1;
92              wait(0.30);
93
94      }
95
96      else if(sev_ch == '1'){
97              ss_a = 1;
98              ss_b = 0;
99              ss_c = 0;
100             ss_d = 1;
101             ss_e = 1;
102             ss_f = 1;
103             ss_g = 1;
104             ss_dot = 1;
105             wait(0.30);
106
107     }
108
109     else if(sev_ch == '2'){
110             ss_a = 0;
111             ss_b = 0;
112             ss_c = 1;
113             ss_d = 0;
114             ss_e = 0;
115             ss_f = 1;
116             ss_g = 0;
117             ss_dot = 1;
118             wait(0.30);
119
120     }
121
122     else if(sev_ch == '3'){
123             ss_a = 0;
124             ss_b = 0;
125             ss_c = 0;
126             ss_d = 0;
127             ss_e = 1;
128             ss_f = 1;
129             ss_g = 0;
130             ss_dot = 1;
131             wait(0.30);
132
133     }
```

```
134     else if(sev_ch == '4'){
135             ss_a = 1;
136             ss_b = 1;
137             ss_c = 0;
138             ss_d = 1;
139             ss_e = 1;
140             ss_f = 0;
141             ss_g = 0;
142             ss_dot = 1;
143             wait(0.30);
144 
145     }
146 
147     else if(sev_ch == '5'){
148             ss_a = 0;
149             ss_b = 1;
150             ss_c = 0;
151             ss_d = 0;
152             ss_e = 1;
153             ss_f = 0;
154             ss_g = 0;
155             ss_dot = 1;
156             wait(0.30);
157 
158     }
159 
160     else if(sev_ch == '6'){
161             ss_a = 0;
162             ss_b = 1;
163             ss_c = 0;
164             ss_d = 0;
165             ss_e = 0;
166             ss_f = 0;
167             ss_g = 0;
168             ss_dot = 1;
169             wait(0.30);
170 
171     }
172 
173     else if(sev_ch == '7'){
174             ss_a = 0;
175             ss_b = 0;
176             ss_c = 0;
177             ss_d = 1;
178             ss_e = 1;
179             ss_f = 1;
180             ss_g = 1;
181             ss_dot = 1;
182             wait(0.30);
183 
```

```
184         }
185
186         else if(sev_ch == '8'){
187                 ss_a = 0;
188                 ss_b = 0;
189                 ss_c = 0;
190                 ss_d = 0;
191                 ss_e = 0;
192                 ss_f = 0;
193                 ss_g = 0;
194                 ss_dot = 1;
195                 wait(0.30);
196
197         }
198
199         else if(sev_ch == '9'){
200                 ss_a = 0;
201                 ss_b = 0;
202                 ss_c = 0;
203                 ss_d = 0;
204                 ss_e = 1;
205                 ss_f = 0;
206                 ss_g = 0;
207                 ss_dot = 1;
208                 wait(0.30);
209
210         }
211
212         else if(sev_ch == 'A'){
213                 ss_a = 0;
214                 ss_b = 0;
215                 ss_c = 0;
216                 ss_d = 1;
217                 ss_e = 0;
218                 ss_f = 0;
219                 ss_g = 0;
220                 ss_dot = 1;
221                 wait(0.30);
222
223         }
224
225         else if(sev_ch == 'B'){
226                 ss_a = 0;
227                 ss_b = 0;
228                 ss_c = 0;
229                 ss_d = 0;
230                 ss_e = 0;
231                 ss_f = 0;
232                 ss_g = 0;
233                 ss_dot = 1;
```

```
234             wait(0.30);
235
236     }
237
238     else if(sev_ch == 'C'){
239             ss_a = 0;
240             ss_b = 1;
241             ss_c = 1;
242             ss_d = 0;
243             ss_e = 0;
244             ss_f = 0;
245             ss_g = 1;
246             ss_dot = 1;
247             wait(0.30);
248
249     }
250
251     else if(sev_ch == 'D'){
252             ss_a = 0;
253             ss_b = 0;
254             ss_c = 0;
255             ss_d = 0;
256             ss_e = 0;
257             ss_f = 0;
258             ss_g = 1;
259             ss_dot = 1;
260             wait(0.30);
261
262     }
263
264     else if(sev_ch == 'E'){
265             ss_a = 0;
266             ss_b = 1;
267             ss_c = 1;
268             ss_d = 0;
269             ss_e = 0;
270             ss_f = 0;
271             ss_g = 0;
272             ss_dot = 0;
273             wait(0.30);
274
275     }
276
277     else if(sev_ch == 'F'){
278             ss_a = 0;
279             ss_b = 1;
280             ss_c = 1;
281             ss_d = 1;
282             ss_e = 0;
283             ss_f = 0;
```

```
284             ss_g = 0;
285             ss_dot = 1;
286             wait(0.30);
287 
288     }
289 
290     else if(sev_ch == 'G'){
291             ss_a = 0;
292             ss_b = 1;
293             ss_c = 0;
294             ss_d = 0;
295             ss_e = 0;
296             ss_f = 0;
297             ss_g = 0;
298             ss_dot = 1;
299             wait(0.30);
300 
301     }
302 
303     else if(sev_ch == 'H'){
304             ss_a = 1;
305             ss_b = 0;
306             ss_c = 0;
307             ss_d = 1;
308             ss_e = 0;
309             ss_f = 0;
310             ss_g = 0;
311             ss_dot = 1;
312             wait(0.30);
313 
314     }
315 
316     else if(sev_ch == 'I'){
317             ss_a = 1;
318             ss_b = 1;
319             ss_c = 1;
320             ss_d = 1;
321             ss_e = 0;
322             ss_f = 0;
323             ss_g = 1;
324             ss_dot = 1;
325             wait(0.30);
326 
327     }
328 
329 
330     else if(sev_ch == 'J'){
331             ss_a = 1;
332             ss_b = 0;
333             ss_c = 0;
```

```
334             ss_d = 0;
335             ss_e = 1;
336             ss_f = 1;
337             ss_g = 1;
338             ss_dot = 1;
339             wait(0.30);
340 
341     }
342 
343     else if(sev_ch == 'K'){
344             ss_a = 1;
345             ss_b = 0;
346             ss_c = 0;
347             ss_d = 1;
348             ss_e = 0;
349             ss_f = 0;
350             ss_g = 0;
351             ss_dot = 1;
352             wait(0.30);
353 
354     }
355 
356     else if(sev_ch == 'L'){
357             ss_a = 1;
358             ss_b = 1;
359             ss_c = 1;
360             ss_d = 0;
361             ss_e = 0;
362             ss_f = 0;
363             ss_g = 1;
364             ss_dot = 1;
365             wait(0.30);
366 
367     }
368 
369 
370     else if(sev_ch == 'M'){
371             ss_a = 1;
372             ss_b = 1;
373             ss_c = 1;
374             ss_d = 1;
375             ss_e = 1;
376             ss_f = 1;
377             ss_g = 1;
378             ss_dot = 1;
379             wait(0.30);
380 
381     }
382 
383     else if(sev_ch == 'N'){
```

```
384             ss_a = 1;
385             ss_b = 1;
386             ss_c = 1;
387             ss_d = 1;
388             ss_e = 1;
389             ss_f = 1;
390             ss_g = 1;
391             ss_dot = 1;
392             wait(0.30);
393
394     }
395
396     else if(sev_ch == 'O'){
397             ss_a = 0;
398             ss_b = 0;
399             ss_c = 0;
400             ss_d = 0;
401             ss_e =0 ;
402             ss_f = 0;
403             ss_g = 1;
404             ss_dot = 1;
405             wait(0.30);
406
407     }
408
409     else if(sev_ch == 'P'){
410             ss_a = 0;
411             ss_b = 0;
412             ss_c = 1;
413             ss_d = 1;
414             ss_e = 0;
415             ss_f = 0;
416             ss_g = 0;
417             ss_dot = 1;
418             wait(0.30);
419
420     }
421
422     else if(sev_ch == 'Q'){
423             ss_a = 0;
424             ss_b = 0;
425             ss_c = 0;
426             ss_d = 0;
427             ss_e = 0;
428             ss_f = 0;
429             ss_g = 0;
430             ss_dot = 0;
431             wait(0.30);
432
433     }
```

```
434
435         else if(sev_ch == 'R'){
436                 ss_a = 0;
437                 ss_b = 0;
438                 ss_c = 0;
439                 ss_d = 1;
440                 ss_e = 0;
441                 ss_f = 0;
442                 ss_g = 0;
443                 ss_dot = 1;
444                 wait(0.30);
445
446         }
447
448         else if(sev_ch == 'S'){
449                 ss_a = 0;
450                 ss_b = 1;
451                 ss_c = 0;
452                 ss_d = 0;
453                 ss_e = 1;
454                 ss_f = 0;
455                 ss_g = 0;
456                 ss_dot = 1;
457                 wait(0.30);
458
459         }
460
461         else if(sev_ch == 'T'){
462                 ss_a = 0;
463                 ss_b = 0;
464                 ss_c = 0;
465                 ss_d = 1;
466                 ss_e = 1;
467                 ss_f = 1;
468                 ss_g = 1;
469                 ss_dot = 1;
470                 wait(0.30);
471
472         }
473
474         else if(sev_ch == 'U'){
475                 ss_a = 1;
476                 ss_b = 0;
477                 ss_c = 0;
478                 ss_d = 0;
479                 ss_e = 0;
480                 ss_f = 0;
481                 ss_g = 1;
482                 ss_dot = 1;
483                 wait(0.30);
```

```
484
485         }
486
487         else if(sev_ch == 'V'){
488                 ss_a = 1;
489                 ss_b = 0;
490                 ss_c = 0;
491                 ss_d = 0;
492                 ss_e = 0;
493                 ss_f = 0;
494                 ss_g = 1;
495                 ss_dot = 1;
496                 wait(0.30);
497
498         }
499                 else if(sev_ch == 'W'){
500                 ss_a = 1;
501                 ss_b = 0;
502                 ss_c = 0;
503                 ss_d = 0;
504                 ss_e = 0;
505                 ss_f = 0;
506                 ss_g = 1;
507                 ss_dot = 1;
508                 wait(0.30);
509
510         }
511
512
513         else if(sev_ch == 'X'){
514                 ss_a = 1;
515                 ss_b = 0;
516                 ss_c = 0;
517                 ss_d = 1;
518                 ss_e = 0;
519                 ss_f = 0;
520                 ss_g = 0;
521                 ss_dot = 1;
522                 wait(0.30);
523
524         }
525
526         else if(sev_ch == 'Y'){
527                 ss_a = 1;
528                 ss_b = 0;
529                 ss_c = 0;
530                 ss_d = 1;
531                 ss_e = 1;
532                 ss_f = 0;
533                 ss_g = 0;
```

```
534             ss_dot = 1;
535             wait(0.30);
536 
537     }
538 
539     else if(sev_ch == 'Z'){
540             ss_a = 0;
541             ss_b = 0;
542             ss_c = 1;
543             ss_d = 0;
544             ss_e = 0;
545             ss_f = 1;
546             ss_g = 0;
547             ss_dot = 1;
548             wait(0.30);
549 
550     }
551 
552             ss_a = 1;
553             ss_b = 1;
554             ss_c = 1;
555             ss_d = 1;
556             ss_e = 1;
557             ss_f = 1;
558             ss_g = 1;
559             ss_dot = 1;
560 }
561 
562 // Function for determing the character after storing the
        dot/dash pattern received
563 inline void find_char()
564 {
565 
566     if(count_ch==0)
567     {
568 
569         return;
570     }
571 
572     else if(count_ch==1)    // If the no of characters
        received is 1
573     {                       // the letter can be either E
         or T
574         if(ch[0]==DOT)
575         {
576         pc.printf("E");     // Print the interpreted
            character on the terminal
577         seven_seg_disp('E');
578         }
579 
```

```
580             else if (ch[0]==DASH)
581             {
582             pc.printf("T");
583             seven_seg_disp('T');
584             }
585         }
586
587         else if(count_ch==2)
588         {
589             if((ch[0]==DOT)&&(ch[1]==DASH))
590             {
591                 pc.printf("A");
592             seven_seg_disp('A');
593             }
594
595             else if((ch[0]==DOT)&&(ch[1]==DOT))
596             {
597                 pc.printf("I");
598             seven_seg_disp('I');
599             }
600
601             else if((ch[0]==DASH)&&(ch[1]==DOT))
602             {
603                 pc.printf("N");
604             seven_seg_disp('N');
605             }
606
607             else if((ch[0]==DASH)&&(ch[1]==DASH))
608             {
609                 pc.printf("M");
610             seven_seg_disp('M');
611             }
612         }
613
614         else if(count_ch==3)
615         {
616             if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT))
617             {
618                 pc.printf("S");
619             seven_seg_disp('S');
620             }
621
622             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT)
                    )
623             {
624                 pc.printf("R");
625             seven_seg_disp('R');
626             }
627
628             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
```

```
            ))
629         {
630             pc.printf("W");
631         seven_seg_disp('W');
632         }
633
634         else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
                )
635         {
636             pc.printf("D");
637         seven_seg_disp('D');
638         }
639
640         else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
                ))
641         {
642             pc.printf("G");
643         seven_seg_disp('G');
644         }
645
646         else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
                ))
647         {
648             pc.printf("K");
649         seven_seg_disp('K');
650         }
651
652         else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
                DASH))
653         {
654             pc.printf("O");
655         seven_seg_disp('O');
656         }
657
658         else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
                )
659         {
660             pc.printf("U");
661         seven_seg_disp('U');
662         }
663
664
665     }
666
667     else if(count_ch==4)
668     {
669         if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch
                [3]==DOT))
670         {
671             pc.printf("H");
```

```
672             seven_seg_disp('H');
673             }
674
675             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT)
                    &&(ch[3]==DOT))
676             {
677                 pc.printf("L");
678                seven_seg_disp('L');
679             }
680
681             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
                    &&(ch[3]==DOT))
682             {
683                 pc.printf("F");
684                 seven_seg_disp('F');
685             }
686
687             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DASH))
688             {
689                 pc.printf("V");
690                 seven_seg_disp('V');
691             }
692
693             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
                    )&&(ch[3]==DOT))
694             {
695                 pc.printf("Z");
696                 seven_seg_disp('Z');
697             }
698
699             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
                    )&&(ch[3]==DOT))
700             {
701                 pc.printf("P");
702                seven_seg_disp('P');
703             }
704
705             else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
                    )&&(ch[3]==DASH))
706             {
707                 pc.printf("J");
708               seven_seg_disp('J');
709             }
710
711             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
                    )&&(ch[3]==DASH))
712             {
713                 pc.printf("Y");
714              seven_seg_disp('Y');
```

```
715             }
716
717             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DASH))
718             {
719                 pc.printf("X");
720              seven_seg_disp('X');
721             }
722
723             else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
                    )&&(ch[3]==DASH))
724             {
725                 pc.printf("Q");
726              seven_seg_disp('Q');
727             }
728
729             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
                    )&&(ch[3]==DOT))
730             {
731                 pc.printf("C");
732              seven_seg_disp('C');
733             }
734
735             else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DOT))
736             {
737                 pc.printf("B");
738              seven_seg_disp('B');
739             }
740         }
741
742         else if(count_ch==5)
743         {
744             if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch
                    [3]==DOT)&&(ch[4]==DOT))
745             {
746                 pc.printf("5");
747                 seven_seg_disp('5');
748
749             }
750
751             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
                    &&(ch[3]==DOT)&&(ch[4]==DASH))
752             {
753                 pc.printf("4");
754                 seven_seg_disp('4');
755
756             }
757
758             else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
```

```
    &&(ch[3]==DASH)&&(ch[4]==DASH))
759         {
760             pc.printf("3");
761             seven_seg_disp('3');
762
763         }
764
765         else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
    &&(ch[3]==DASH)&&(ch[4]==DASH))
766         {
767             pc.printf("2");
768             seven_seg_disp('2');
769
770         }
771
772         else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
    )&&(ch[3]==DASH)&&(ch[4]==DASH))
773         {
774             pc.printf("1");
775             seven_seg_disp('1');
776         }
777
778         else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
    &&(ch[3]==DOT)&&(ch[4]==DOT))
779         {
780             pc.printf("6");
781         }
782
783         else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
    )&&(ch[3]==DOT)&&(ch[4]==DOT))
784         {
785             pc.printf("7");
786         }
787
788         else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
    DASH)&&(ch[3]==DOT)&&(ch[4]==DOT))
789         {
790             pc.printf("8");
791         }
792
793         else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
    DASH)&&(ch[3]==DASH)&&(ch[4]==DOT))
794         {
795             pc.printf("9");
796         }
797
798         else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
    DASH)&&(ch[3]==DASH)&&(ch[4]==DASH))
799         {
800             pc.printf("0");
```

```
801                 seven_seg_disp('0');
802             }
803 
804         }
805 
806 
807 }
808 
809 // Counter for no of spaces
810 int count_space = 0;
811 bool bool_space = false;     // Flag for displaying space
812 
813 // Function for handling space
814 inline void space()
815 {
816     ss_a = 0;
817     wait(0.05);
818     ss_a = 1;
819 
820     count_space++;  // Increment the no of spaces
        received
821 
822     if(count_space<3){  // If there are two between
        letters then interpret the letter
823 
824     // depending on count, determine the character
825     find_char();
826 
827     count_ch = 0;    // Reset the count and array
828     ch[0] = 0;
829     ch[1] = 0;
830     ch[2] = 0;
831     ch[3] = 0;
832     ch[4] = 0;
833     bool_space = false;
834     }
835 
836     else if ((count_space>=5)&&(bool_space==false)) //
        Display space
837     {
838      bool_space = true;
839 
840     pc.printf("space");
841 
842 
843     }
844 
845 }
846 
847 // Function for handling dot
```

```
848 inline void dot()
849 {
850     count_space = 0;
851     ss_d = 0;
852     wait(0.05);
853     ss_d = 1;
854 
855     buzz=1;
856     wait(0.05);
857     buzz = 0;
858     if(count_ch>=5)
859     {
860     space();
861     }
862     else
863     {
864         ch[count_ch] = DOT;
865         count_ch++;
866     }
867 }
868 
869 // Function for handling dash
870 inline void dash()
871 {
872     count_space = 0;
873     ss_g = 0;
874     wait(0.05);
875     ss_g = 1;
876 
877     buzz=1;
878     wait(0.15);
879     buzz = 0;
880 
881     if(count_ch>=5)
882     {
883     space();
884     }
885     else
886     {
887         ch[count_ch] = DASH;
888         count_ch++;
889     }}
890 
891 // Function for turning off the 7 segment display
892 void segment_off()
893 {
894     ss_a = 1;
895     ss_b = 1;
896     ss_c = 1;
897     ss_d = 1;
```

```
898         ss_e = 1;
899         ss_f = 1;
900         ss_g = 1;
901         ss_dot = 1;
902 
903     }
904 
905     // Function for turning on the 7 segment display
906     void segment_on()
907     {
908         ss_a = 0;
909         ss_b = 0;
910         ss_c = 0;
911         ss_d = 0;
912         ss_e = 0;
913         ss_f = 0;
914         ss_g = 0;
915         ss_dot = 0;
916     }
917 
918     // Debugging function
919     void key_rise_int1()
920     {
921         wait(0.01);
922         if(key)
923         {
924             pc.printf("pressed");
925         }
926 
927     }
928 
929     // Debugging function
930     void key_fall_int1()
931     {
932         wait(0.01);
933         if(!key)
934         {
935             pc.printf("released");
936 
937         }
938     }
939 
940     // Interrupt handler for key press
941     void key_rise_int()
942     {
943 
944         wait(0.01);
945         //read the value of the key after 10ms and set the
                flag
946         if(key)
```

```
947     {
948         timer_400.stop();    // Stop the space timer
949         flag = true;
950         //start a timer
951         timer_key.start();  // Start the button press
                timer
952        // pc.printf("key pressed");
953     }
954 
955     else
956     {
957         flag = false;
958     }
959 
960 }
961 
962 // Interrupt handler for key release
963 void key_fall_int()
964 {
965     wait(0.01);
966 
967     if(!key)
968     {
969         //pc.printf("key released");
970         //timer_400.reset();
971         // stop the timer
972         timer_key.stop();
973 
974         // read the timer value and decide if dot, dash
975         int timer_key_val = timer_key.read_ms();
976 
977         if((timer_key_val>40)&&(timer_key_val<220)&&(flag
                ==true) )
978         {  // dot received
979             /*myled1 = 1;
980             wait(0.05);
981             myled1 = 0;*/
982             dot();
983         }
984 
985         // dash received
986         else if (timer_key_val>220){
987             /*myled2 = 1;
988             wait(0.05);
989             myled2 = 0;*/
990             dash();
991         }
992 
993         // reset the timer
994         timer_key.stop();
```

```
995             timer_key.reset();
996             flag=0;
997             timer_400.reset();  // Reset and restart the
                    space timer
998             timer_400.start();
999
1000        }
1001    }
1002
1003
1004
1005    int main() {
1006
1007
1008        // Check the leds
1009        myled1 = 0;
1010        myled2 = 1;
1011        myled3 = 1;
1012        myled4 = 1;
1013
1014        wait(0.5);
1015        myled1 = 0;
1016        myled2 = 0;
1017        myled3 = 0;
1018        myled4 = 0;
1019
1020        // Check the seven segment
1021        segment_off();
1022        wait(0.5);
1023        segment_on();
1024        wait(0.5);
1025        segment_off();
1026
1027        pc.baud(9600);
1028        row = 1;
1029
1030
1031        // start the timer for 400ms
1032        timer_400.start();
1033
1034        key.rise(&key_rise_int);     // Set up the interrup
                handler
1035        key.fall(&key_fall_int);
1036
1037
1038        while(1) {
1039            // if timer value hits 400 ms, then reset the
                    timer and read as space.
1040            if(timer_400.read_ms()>=400)
1041            {
```

```
1042            timer_400.reset();
1043
1044            if(!flag)
1045            {
1046            //pc.printf("Got a space\n");
1047            /*myled3 = 1;
1048            wait(0.05);
1049            myled3 = 0;*/
1050
1051                 space();
1052
1053            }
1054            }
1055        }
1056    }
```

Difference between Polling and Interrupt driven programs

Polling and interrupts based detections are two common ways to check if a certain key has been pressed or if a certain pin has been either pulled up or pulled down. In a polling driven program, the CPU wastes cycles while waiting for the pin to go high or low, this happens because the code if put in a infinite while loop with an if statement reading the pin configuration. In an interrupt driven program, the CPU is generally interrupted when a certain pin reaches a particular condition and thus with this the micro-controller doesn't keep checking the status of a pin unneccessarily.