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,

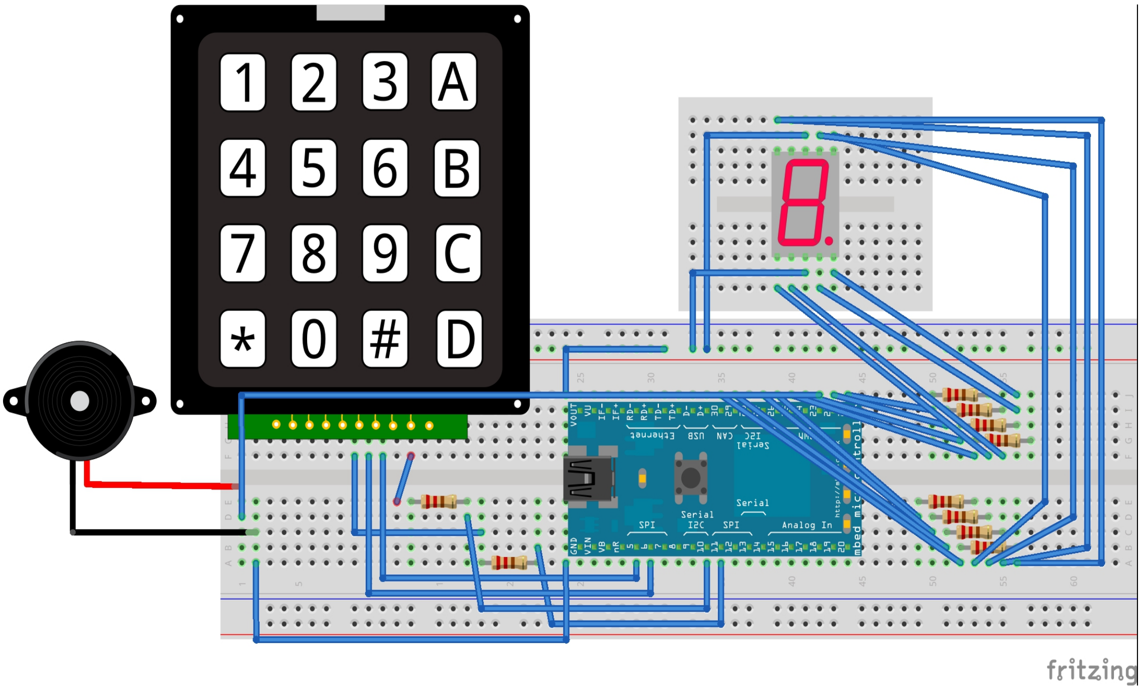

Figure 1: Hardware Connections

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
    pc.printf("Bits are %d %d %d %d\n",a[0],a[1],a[2],a[3]);

}

int main() {

  // serial speed
  pc.baud(9600);


  // 18 second timer period
  ticker_period_18.attach(&get_adc,18.0);
  get_adc();
  while(1) {

    // led1
    myled1 = 1;
    wait(0.1*a[0]);
    myled1 = 0;

    //led2
    myled2 = 1;
    wait(0.1*a[1]);
    myled2 = 0;

    //led3
    myled3 = 1;
    wait(0.1*a[2]);
    myled3 = 0;

    //led4
    myled4 = 1;
    wait(0.1*a[3]);
    myled4 = 0;

  }

}
```

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]);

}

int main() {

  // serial speed
  pc.baud(9600);


  // 18 second timer period
  ticker_period_18.attach(&get_adc,18.0);
  get_adc();
  while(1) {

    // led1
    myled1 = 1;
    wait(0.1*a[0]);
    myled1 = 0;

    //led2
    myled2 = 1;
    wait(0.1*a[1]);
    myled2 = 0;

    //led3
    myled3 = 1;
    wait(0.1*a[2]);
    myled3 = 0;

    //led4
    myled4 = 1;
    wait(0.1*a[3]);
    myled4 = 0;

  }

}
```

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,

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

```

Idea : Using the keypad, interrupt when the key is pressed.

*/

#include ”mbed.h”


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


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

//define the pins for # and 2
DigitalOut pin1(p10);
DigitalOut pin2(p11);


// keypad association
DigitalIn col4(p5);
InterruptIn col3(p6);
InterruptIn col2(p7);
DigitalIn col1(p8);

// ISR when button pressed - lights up led1
void light0(){
        myled1 = 1;
        wait(0.2);
        myled1 = 0;
        wait(0.2);
    }

// ISR when button pressed - lights up led2
void light1(){
        myled2 = 1;
        wait(0.2);
        myled2 = 0;
        wait(0.2);
}

int main() {

    pin1 = 1 ;
    pin2 = 1 ;

    // associate interrupts with functions light1 and light0
    col3.rise(&light1);
    col2.rise(&light0);
    while(1) {
    }
}
```

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,

```
/*
ese519-lab1-part3
    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

*/

#include "mbed.h"


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

bool flag = false;

```

```
// Interrupt whenever pin is pressed (#) - keypad
InterruptIn key(p6);
DigitalOut row(p11);

// led interface - visual feedback
DigitalOut myled1(LED1);
DigitalOut myled2(LED2);
DigitalOut myled3(LED3);
DigitalOut myled4(LED4);

/*
*   Dot represented by led1
*   Dash represented by led2
*   Space represented by led3
*/

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

// timers to ensure dot and dash
Timer timer_400;
Timer timer_key;

// ISR when the key is just pressed on the keypad (for debugging)
void key_rise_int1()
{
    // debouncing code - wait and check
    wait(0.01);
    if(key)
    {
        // serial debug
        pc.printf(”pressed”);
    }

}

// ISR when the key is just pressed on the keypad (for debugging)
void key_fall_int1()
{
    // debouncing code - wait and check
    wait(0.01);
    if(!key)
    {
        // serial debug
        pc.printf(”released”);

    }
}
```

```


// ISR when the key is just pressed on the keypad
void key_rise_int()
{
    // debouncing code - wait and check
    //read the value of the key after 10ms and set the flag
    wait(0.01);

    if(key)
    {
        timer_400.stop();   // Stop the space timer
        flag = true;
        //start a timer
        timer_key.start();  // Start counting the duration of the key press
        pc.printf(”key pressed”);   // For debugging
    }

    else
    {
        flag = false;
    }

}

// ISR when the key is just pressed on the keypad
void key_fall_int()
{
    wait(0.01);

    if(!key)
    {
        pc.printf(”key released”);  // For debugging
        //timer_400.reset();
        // stop the timer
        timer_key.stop();   // Stop the first timer which measures the key press

        // read the timer value and decide if dot, dash
        int timer_key_val = timer_key.read_ms();

        if((timer_key_val >40)&&(timer_key_val <220)&&(flag==true) )  // Dot has been pressed
        {
            myled1 = 1;
            wait(0.05);
            myled1 = 0;
        }

        else if (timer_key_val>220){     // Dash has been pressed
            myled2 = 1;
            wait(0.05);
            myled2 = 0;
        }

        // Reset and stop the measurement timer
        timer_key.stop();
        timer_key.reset();
        flag=0;
        timer_400.reset();  // Reset and restart the timer for space
        timer_400.start();

    }
}


int main() {

    myled1 = 0;      // blink the leds once to indicate start of the program
    myled2 = 1;
    myled3 = 1;
    myled4 = 1;

    wait(0.5);
    myled1 = 0;
    myled2 = 0;
    myled3 = 0;
    myled4 = 0;

    pc.baud(9600);
    row = 1;


    // start the timer for 400ms to check the occurence of space
    timer_400.start();

    key.rise(&key_rise_int);     // Set up the button interrupt handlers
    key.fall(&key_fall_int);


    while(1) {
        // if timer value hits 400 ms, then reset the timer and read as space.
        if(timer_400.read_ms()>=400)
        {
        timer_400.reset();
        if(!flag)
        {
        //pc.printf("Got a space\n");
        myled3 = 1;     //Indicate the presence of space by blinking led
        wait(0.05);
        myled3 = 0;
        }
        }
    }
}
```

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:

```
#include "mbed.h"
/*
ese519-lab1-part4
    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

Dot - Segment D
Dash - Segment G
Space - Segment A
*/

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


bool flag = false;

// Define the row and column for keypad
InterruptIn key(p6);
DigitalOut row(p11);

DigitalOut myled1(LED1);
DigitalOut myled2(LED2);
DigitalOut myled3(LED3);
DigitalOut myled4(LED4);
Serial pc(USBTX, USBRX); // tx, rx

// 2 timers used for space and dot/dash measurement
Timer timer_400;
Timer timer_key;

// seven segment display interface
// Mbed - Seven Segment (common anode)
// p21 - 1
// p22 - 2
// gnd - 3
// p24 - 4
// p25 - 5
// p26 - 6
// p27 - 7
// gnd - 8
// p29 - 9
// p30 - 10

DigitalOut ss_a(p27);
DigitalOut ss_b(p26);
DigitalOut ss_c(p24);
DigitalOut ss_d(p22);
DigitalOut ss_e(p21);
DigitalOut ss_f(p29);
DigitalOut ss_g(p30);
DigitalOut ss_dot(p25);

// Function for handling dot behavior
inline void dot()
{
ss_d = 0;
wait(0.05);
ss_d = 1;

}

// Function for handling dash behavior
inline void dash()
{
    ss_g = 0;
    wait(0.05);
    ss_g = 1;

}

// Function for handling space behavior
inline void space()
{
    ss_a = 0;
    wait(0.05);
    ss_a = 1;

}
// Function for turning of the 7 segment display

void segment_off()
{
    ss_a = 1;
    ss_b = 1;
    ss_c = 1;
    ss_d = 1;
    ss_e = 1;
    ss_f = 1;
    ss_g = 1;
    ss_dot = 1;

}

// Function for turning on all leds in the 7 segment
   display
// Used for debugging
void segment_on()
{
    ss_a = 0;
    ss_b = 0;
    ss_c = 0;
    ss_d = 0;
    ss_e = 0;
    ss_f = 0;
    ss_g = 0;
    ss_dot = 0;
}

```

```
// Interrupt handler for key press - Debugging purpose
void key_rise_int1()
{
    wait(0.01); // Debounce period
    if(key)
    {
        pc.printf(”pressed”);
    }

}

// Interrupt handler for key release - Debugging purpose
void key_fall_int1()
{
    wait(0.01);
    if(!key)
    {
        pc.printf(”released”);

    }
}

// Interrupt handler for key press
void key_rise_int()
{

    wait(0.01);
    //read the value of the key after 10ms and set the flag
    if(key)
    {
        timer_400.stop();   // Stop the space timer
        flag = true;
        //start a timer for measuring the duration of the key press
        timer_key.start();
        pc.printf(”key pressed”);
    }

    else
    {
        flag = false;    // Debouncing failed- key wasn't pressed
    }

}

// Interrupt handler for key release
void key_fall_int()
{
    wait(0.01);

    if(!key)
    {
        pc.printf("key released");
        //timer_400.reset();
        // stop the timer used for measuring key press
        timer_key.stop();

        // read the timer value and decide if dot, dash
        int timer_key_val = timer_key.read_ms();

        if((timer_key_val>40)&&(timer_key_val<220)&&(flag==true) )
        {
            /*myled1 = 1;
            wait(0.05);
            myled1 = 0;*/
            dot();  // Dot pressed
        }

        else if (timer_key_val>220){
            /*myled2 = 1;
            wait(0.05);
            myled2 = 0;*/
            dash(); //Dash pressed
        }

        // reset the measurement timer
        timer_key.stop();
        timer_key.reset();
        flag=0;
        timer_400.reset();  // Restart the space timer after reset
        timer_400.start();

    }
}


int main() {


    // Check the leds
    myled1 = 0;
    myled2 = 1;
    myled3 = 1;
    myled4 = 1;

    wait(0.5);
    myled1 = 0;
    myled2 = 0;
    myled3 = 0;
    myled4 = 0;

    // Check the seven segment
    segment_off();
    wait(0.5);
    segment_on();
    wait(0.5);
    segment_off();

    pc.baud(9600);
    row = 1;


    // start the timer for 400ms
    timer_400.start();

    key.rise(&key_rise_int);
    key.fall(&key_fall_int);    // Set up the interrupt handlers


    while(1) {
        // if timer value hits 400 ms, then reset the timer and read as space.
        if(timer_400.read_ms()>=400)
        {
        timer_400.reset();

        if(!flag)
        {
        //pc.printf(”Got a space\n”);
        /*myled3 = 1;
        wait(0.05);
        myled3 = 0;*/
        space();    // Space occured
        }
        }
    }
}
```

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);
DigitalOut myled3(LED3);
DigitalOut myled4(LED4);
Serial pc(USBTX, USBRX); // tx, rx

// Timers used for detecting dot/dash and space
Timer timer_400;    // Timer for space
Timer timer_key;

// seven segment display
// Mbed - Seven Segment (common anode)
// p21 - 1
// p22 - 2
// gnd - 3
// p24 - 4
// p25 - 5
// p26 - 6
// p27 - 7
// gnd - 8
// p29 - 9
// p30 - 10
DigitalOut ss_a(p27);
DigitalOut ss_b(p26);
DigitalOut ss_c(p24);
DigitalOut ss_d(p22);
DigitalOut ss_e(p21);
DigitalOut ss_f(p29);
DigitalOut ss_g(p30);
DigitalOut ss_dot(p25);

// Function for determing the character after storing the dot/dash pattern received
inline void find_char()
{

    if(count_ch==0)
    {
        return;
    }

    else if(count_ch==1)    // If the no of characters received is 1
    {                        // the letter can be either E or T
        if(ch[0]==DOT)
        {
        pc.printf("E");    // Print the interpreted character on the terminal
        }

        else if (ch[0]==DASH)
        {
        pc.printf("T");
        }
    }

    else if(count_ch==2)
    {
        if((ch[0]==DOT)&&(ch[1]==DASH))
        {
            pc.printf("A");
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT))
        {
            pc.printf("I");
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT))
        {
            pc.printf("N");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH))
        {
            pc.printf("M");
        }
    }

    else if(count_ch==3)
    {
        if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT))
        {
            pc.printf("S");
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT))
        {
            pc.printf("R");
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH))
        {
            pc.printf("W");
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT))
        {
            pc.printf(”D”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT))
        {
            pc.printf(”G”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH))
        {
            pc.printf(”K”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DASH))
        {
            pc.printf(”O”);
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH))
        {
            pc.printf(”U”);
        }


    }

    else if(count_ch==4)
    {
        if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DOT))
        {
            pc.printf(”H”);
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT)&&(ch[3]==DOT))
        {
            pc.printf(”L”);
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)&&(ch[3]==DOT))
        {
            pc.printf(”F”);
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
            &&(ch[3]==DASH))
        {
            pc.printf("V");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
            )&&(ch[3]==DOT))
        {
            pc.printf("Z");
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
            )&&(ch[3]==DOT))
        {
            pc.printf("P");
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
            )&&(ch[3]==DASH))
        {
            pc.printf("J");
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
            )&&(ch[3]==DASH))
        {
            pc.printf("Y");
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
            &&(ch[3]==DASH))
        {
            pc.printf("X");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
            )&&(ch[3]==DASH))
        {
            pc.printf("Q");
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
            )&&(ch[3]==DOT))
        {
            pc.printf("C");
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
            &&(ch[3]==DOT))
        {
            pc.printf("B");
        }
    }

    else if(count_ch==5)
    {
        if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("5");
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DOT)&&(ch[4]==DASH))
        {
            pc.printf("4");
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("3");
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)&&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("2");
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH)&&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("1");
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("6");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT)&&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("7");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
            DASH)&&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("8");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DASH)&&(ch[3]==DASH)&&(ch[4]==DOT))
        {
            pc.printf("9");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DASH)&&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("0");
        }

    }


}

// Function for space behavior
inline void space()
{
    ss_a = 0;
    wait(0.05);
    ss_a = 1;

    // depending on count, determine the character
    find_char();

    count_ch = 0;   // Reset the number of characters count
    ch[0] = 0;      // Clear the array
    ch[1] = 0;
    ch[2] = 0;
    ch[3] = 0;
    ch[4] = 0;

}

inline void dot()
{
    ss_d = 0;
    wait(0.05);
    ss_d = 1;
    if(count_ch>=5)
    {
    space();   // When count >= 5 the array must be
```

```
        checked for the ascii character
    }
    else
    {
        ch[count_ch] = DOT;
        count_ch++;
    }
}

inline void dash()
{
    ss_g = 0;
    wait(0.05);
    ss_g = 1;

    if(count_ch>=5) // When count >= 5 the array must be
        checked for the ascii character
    {
    space();
    }
    else
    {
        ch[count_ch] = DASH;
        count_ch++;
    }}

// function to turn off the 7 segment
void segment_off()
{
    ss_a = 1;
    ss_b = 1;
    ss_c = 1;
    ss_d = 1;
    ss_e = 1;
    ss_f = 1;
    ss_g = 1;
    ss_dot = 1;

}
// function to turn on the 7 segment
void segment_on()
{
    ss_a = 0;
    ss_b = 0;
    ss_c = 0;
    ss_d = 0;
    ss_e = 0;
    ss_f = 0;
    ss_g = 0;
    ss_dot = 0;
```

```
}

// debugging function
void key_rise_int1()
{
    wait(0.01);
    if(key)
    {
        pc.printf("pressed");
    }

}

// debugging function
void key_fall_int1()
{
    wait(0.01);
    if(!key)
    {
        pc.printf("released");

    }
}

// Interrupt handler for key press
void key_rise_int()
{

    wait(0.01);
    //read the value of the key after 10ms and set the flag
    if(key)
    {
        timer_400.stop();
        flag = true;
        //start a timer
        timer_key.start();
       // pc.printf("key pressed");
    }

    else
    {
        flag = false;
    }

}
// Interrupt handler for key press
void key_fall_int()
{
    wait(0.01);

    if(!key)
    {
        //pc.printf("key released");
        //timer_400.reset();
        // stop the timer
        timer_key.stop();

        // read the timer value and decide if dot, dash
        int timer_key_val = timer_key.read_ms();

        if((timer_key_val>40)&&(timer_key_val<220)&&(flag==true) )
        {
            /*myled1 = 1;
            wait(0.05);
            myled1 = 0;*/
            dot();
        }

        else if (timer_key_val>220){
            /*myled2 = 1;
            wait(0.05);
            myled2 = 0;*/
            dash();
        }

        // reset the timer
        timer_key.stop();   // Stop and reset the dot/dash timer
        timer_key.reset();
        flag=0;
        timer_400.reset(); // Reset the space timer
        timer_400.start();

    }
}


int main() {


    // Check the leds
    myled1 = 0;
    myled2 = 1;
    myled3 = 1;
    myled4 = 1;

    wait(0.5);
    myled1 = 0;
    myled2 = 0;
    myled3 = 0;
    myled4 = 0;

    // Check the seven segment
    segment_off();
    wait(0.5);
    segment_on();
    wait(0.5);
    segment_off();

    pc.baud(9600);
    row = 1;


    // start the timer for 400ms
    timer_400.start();

    key.rise(&key_rise_int);    // Setup the interrupt handlers
    key.fall(&key_fall_int);


    while(1) {
        // if timer value hits 400 ms, then reset the timer and read as space.
        if(timer_400.read_ms()>=400)
        {
        timer_400.reset();

        if(!flag)
        {
        //pc.printf(”Got a space\n”);
        /*myled3 = 1;
        wait(0.05);
        myled3 = 0;*/
        space();
        }
        }
    }
}
```

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:

```
#include "mbed.h"

#define DOT 1
#define DASH 2
/*
ese519-lab1-Extra Credit Part 1
    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
Buzzer is interfaced to beep when a dot is pressed and a longer beep
when a dash is pressed

*/

// 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);
DigitalOut myled3(LED3);
DigitalOut myled4(LED4);
Serial pc(USBTX, USBRX); // tx, rx

// Timers used for detecting dot/dash and space
Timer timer_400;     // Timer for space
Timer timer_key;

// seven segment display
// Mbed - Seven Segment (common anode)
// p21 - 1
// p22 - 2
// gnd - 3
// p24 - 4
// p25 - 5
// p26 - 6
// p27 - 7
// gnd - 8
// p29 - 9
// p30 - 10
DigitalOut ss_a(p27);
DigitalOut ss_b(p26);
DigitalOut ss_c(p24);
DigitalOut ss_d(p22);
DigitalOut ss_e(p21);
DigitalOut ss_f(p29);
DigitalOut ss_g(p30);
DigitalOut ss_dot(p25);

//buzzer
DigitalOut buzz(p23);

// Function for determing the character after storing the dot/dash pattern received
inline void find_char()
{

    if(count_ch==0)
    {
        return;
    }

    else if(count_ch==1)    // If the no of characters received is 1
    {                       // the letter can be either E or T
        if(ch[0]==DOT)
        {
        pc.printf("E");     // Print the interpreted character on the terminal
        }

        else if (ch[0]==DASH)
        {
        pc.printf("T");
        }
    }

    else if(count_ch==2)
    {
        if((ch[0]==DOT)&&(ch[1]==DASH))
        {
            pc.printf("A");
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT))
        {
            pc.printf("I");
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT))
        {
            pc.printf("N");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH))
        {
            pc.printf("M");
        }
    }

    else if(count_ch==3)
    {
        if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT))
        {
            pc.printf("S");
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT)
            )
        {
            pc.printf("R");
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
            ))
        {
            pc.printf("W");
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
            )
        {
            pc.printf(”D”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
            ))
        {
            pc.printf(”G”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
            ))
        {
            pc.printf(”K”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
           DASH))
        {
            pc.printf(”O”);
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
            )
        {
            pc.printf(”U”);
        }


    }

    else if(count_ch==4)
    {
        if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch
            [3]==DOT))
        {
            pc.printf(”H”);
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT)
           &&(ch[3]==DOT))
        {
            pc.printf(”L”);
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
           &&(ch[3]==DOT))
        {
            pc.printf(”F”);
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
            &&(ch[3]==DASH))
        {
            pc.printf(”V”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
            )&&(ch[3]==DOT))
        {
            pc.printf(”Z”);
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
            )&&(ch[3]==DOT))
        {
            pc.printf(”P”);
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
            )&&(ch[3]==DASH))
        {
            pc.printf(”J”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
            )&&(ch[3]==DASH))
        {
            pc.printf(”Y”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
            &&(ch[3]==DASH))
        {
            pc.printf(”X”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
            )&&(ch[3]==DASH))
        {
            pc.printf(”Q”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
            )&&(ch[3]==DOT))
        {
            pc.printf(”C”);
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DOT))
        {
            pc.printf("B");
        }
    }

    else if(count_ch==5)
    {
        if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("5");
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DOT)&&(ch[4]==DASH))
        {
            pc.printf("4");
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("3");
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)&&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("2");
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH)&&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("1");
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("6");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT)&&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("7");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DASH)&&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("8");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DASH)&&(ch[3]==DASH)&&(ch[4]==DOT))
        {
            pc.printf("9");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DASH)&&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("0");
        }

    }


}


inline void space()
{
    ss_a = 0;
    wait(0.05);
    ss_a = 1;


    // depending on count, determine the character
    find_char();

    count_ch = 0;
    ch[0] = 0;
    ch[1] = 0;
    ch[2] = 0;
    ch[3] = 0;
    ch[4] = 0;

}

inline void dot()
{
    ss_d = 0;
    wait(0.05);
    ss_d = 1;


    buzz = 1;
    wait(0.05);
    buzz = 0;

    if(count_ch>=5)
    {
    space();    // When count >= 5 the array must be checked for the ascii character
    }
    else
    {
        ch[count_ch] = DOT;
        count_ch++;
    }
}

inline void dash()
{
    ss_g = 0;
    wait(0.05);
    ss_g = 1;

    buzz = 1;
    wait(0.15);
    buzz = 0;

    if(count_ch>=5)  // When count >= 5 the array must be checked for the ascii character
    {
    space();
    }
    else
    {
        ch[count_ch] = DASH;
        count_ch++;
    }}

// function to turn off the 7 segment
void segment_off()
{
    ss_a = 1;
    ss_b = 1;
    ss_c = 1;
    ss_d = 1;
    ss_e = 1;
    ss_f = 1;
    ss_g = 1;
    ss_dot = 1;

```

```
}

// function to turn on the 7 segment
void segment_on()
{
    ss_a = 0;
    ss_b = 0;
    ss_c = 0;
    ss_d = 0;
    ss_e = 0;
    ss_f = 0;
    ss_g = 0;
    ss_dot = 0;
}

// debugging function
void key_rise_int1()
{
    wait(0.01);
    if(key)
    {
        pc.printf("pressed");
    }

}

// debugging function
void key_fall_int1()
{
    wait(0.01);
    if(!key)
    {
        pc.printf("released");

    }
}

// Interrupt handler for key press
void key_rise_int()
{

    wait(0.01);
    //read the value of the key after 10ms and set the flag
    if(key)
    {
        timer_400.stop();
        flag = true;
        //start a timer
        timer_key.start();
        // pc.printf("key pressed");
    }

    else
    {
        flag = false;
    }

}
// Interrupt handler for key release
void key_fall_int()
{
    wait(0.01);

    if(!key)
    {
        //pc.printf("key released");
        //timer_400.reset();
        // stop the timer
        timer_key.stop();

        // read the timer value and decide if dot, dash
        int timer_key_val = timer_key.read_ms();

        if((timer_key_val>40)&&(timer_key_val<220)&&(flag
            ==true) )
        {
            /*myled1 = 1;
            wait(0.05);
            myled1 = 0;*/
            dot();
        }

        else if (timer_key_val>220){
            /*myled2 = 1;
            wait(0.05);
            myled2 = 0;*/
            dash();
        }

        // reset the timer
        timer_key.stop();
        timer_key.reset();
        flag=0;
        timer_400.reset();  // Reset the space timer
        timer_400.start();

    }
}

```

```

int main() {


    // Check the leds
    myled1 = 0;
    myled2 = 1;
    myled3 = 1;
    myled4 = 1;

    wait(0.5);
    myled1 = 0;
    myled2 = 0;
    myled3 = 0;
    myled4 = 0;

    // Check the seven segment
    segment_off();
    wait(0.5);
    segment_on();
    wait(0.5);
    segment_off();

    pc.baud(9600);
    row = 1;


    // start the timer for 400ms
    timer_400.start();

    key.rise(&key_rise_int); // Setup the interrupt handlers
    key.fall(&key_fall_int);


    while(1) {
        // if timer value hits 400 ms, then reset the timer and read as space.
        if(timer_400.read_ms()>=400)
        {
        timer_400.reset();

        if(!flag)
        {
        //pc.printf(”Got a space\n”);

        space();
        }
        }
    }
}
```

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:

```
#include "mbed.h"

#define DOT 1
#define DASH 2
/*
ese519-lab1-Extra Credit Seven segment
    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

Buzzer is interfaced to beep when a dot is pressed and a longer beep
when a dash is pressed

ASCII characters are display on the 7 segment display

2 spaces are allowed between letters.
5 or more spaces is considered as one space

*/

// 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

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);

DigitalOut myled1(LED1);
DigitalOut myled2(LED2);
DigitalOut myled3(LED3);
DigitalOut myled4(LED4);
Serial pc(USBTX, USBRX); // tx, rx

// Timers used for detecting dot/dash and space
Timer timer_400;    // Timer for space
Timer timer_key;

DigitalOut buzz(p23);

// seven segment display
// Mbed - Seven Segment (common cathode)
// p21 - 1
// p22 - 2
// gnd - 3
// p24 - 4
// p25 - 5
// p26 - 6
// p27 - 7
// gnd - 8
// p29 - 9
// p30 - 10
DigitalOut ss_a(p27);
DigitalOut ss_b(p26);
DigitalOut ss_c(p24);
DigitalOut ss_d(p22);
DigitalOut ss_e(p21);
DigitalOut ss_f(p29);
DigitalOut ss_g(p30);
DigitalOut ss_dot(p25);

// Function which displays ASCII character on the 7 segment display given a character
void seven_seg_disp(char sev_ch)
{
    if(sev_ch == '0'){
```

```
                ss_a = 0;
                ss_b = 0;
                ss_c = 0;
                ss_d = 0;
                ss_e = 0;
                ss_f = 0;
                ss_g = 1;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == '1'){
                ss_a = 1;
                ss_b = 0;
                ss_c = 0;
                ss_d = 1;
                ss_e = 1;
                ss_f = 1;
                ss_g = 1;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == '2'){
                ss_a = 0;
                ss_b = 0;
                ss_c = 1;
                ss_d = 0;
                ss_e = 0;
                ss_f = 1;
                ss_g = 0;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == '3'){
                ss_a = 0;
                ss_b = 0;
                ss_c = 0;
                ss_d = 0;
                ss_e = 1;
                ss_f = 1;
                ss_g = 0;
                ss_dot = 1;
                wait(0.30);

        }
        else if(sev_ch == '4'){
                ss_a = 1;
                ss_b = 1;
                ss_c = 0;
                ss_d = 1;
                ss_e = 1;
                ss_f = 0;
                ss_g = 0;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == '5'){
                ss_a = 0;
                ss_b = 1;
                ss_c = 0;
                ss_d = 0;
                ss_e = 1;
                ss_f = 0;
                ss_g = 0;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == '6'){
                ss_a = 0;
                ss_b = 1;
                ss_c = 0;
                ss_d = 0;
                ss_e = 0;
                ss_f = 0;
                ss_g = 0;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == '7'){
                ss_a = 0;
                ss_b = 0;
                ss_c = 0;
                ss_d = 1;
                ss_e = 1;
                ss_f = 1;
                ss_g = 1;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == '8'){
                ss_a = 0;
                ss_b = 0;
                ss_c = 0;
                ss_d = 0;
                ss_e = 0;
                ss_f = 0;
                ss_g = 0;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == '9'){
                ss_a = 0;
                ss_b = 0;
                ss_c = 0;
                ss_d = 0;
                ss_e = 1;
                ss_f = 0;
                ss_g = 0;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == 'A'){
                ss_a = 0;
                ss_b = 0;
                ss_c = 0;
                ss_d = 1;
                ss_e = 0;
                ss_f = 0;
                ss_g = 0;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == 'B'){
                ss_a = 0;
                ss_b = 0;
                ss_c = 0;
                ss_d = 0;
                ss_e = 0;
                ss_f = 0;
                ss_g = 0;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == 'C'){
                ss_a = 0;
                ss_b = 1;
                ss_c = 1;
                ss_d = 0;
                ss_e = 0;
                ss_f = 0;
                ss_g = 1;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == 'D'){
                ss_a = 0;
                ss_b = 0;
                ss_c = 0;
                ss_d = 0;
                ss_e = 0;
                ss_f = 0;
                ss_g = 1;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == 'E'){
                ss_a = 0;
                ss_b = 1;
                ss_c = 1;
                ss_d = 0;
                ss_e = 0;
                ss_f = 0;
                ss_g = 0;
                ss_dot = 0;
                wait(0.30);

        }

        else if(sev_ch == 'F'){
                ss_a = 0;
                ss_b = 1;
                ss_c = 1;
                ss_d = 1;
                ss_e = 0;
                ss_f = 0;
```

```
            ss_g = 0;
            ss_dot = 1;
            wait(0.30);

    }

    else if(sev_ch == 'G'){
            ss_a = 0;
            ss_b = 1;
            ss_c = 0;
            ss_d = 0;
            ss_e = 0;
            ss_f = 0;
            ss_g = 0;
            ss_dot = 1;
            wait(0.30);

    }

    else if(sev_ch == 'H'){
            ss_a = 1;
            ss_b = 0;
            ss_c = 0;
            ss_d = 1;
            ss_e = 0;
            ss_f = 0;
            ss_g = 0;
            ss_dot = 1;
            wait(0.30);

    }

    else if(sev_ch == 'I'){
            ss_a = 1;
            ss_b = 1;
            ss_c = 1;
            ss_d = 1;
            ss_e = 0;
            ss_f = 0;
            ss_g = 1;
            ss_dot = 1;
            wait(0.30);

    }


    else if(sev_ch == 'J'){
            ss_a = 1;
            ss_b = 0;
            ss_c = 0;
            ss_d = 0;
            ss_e = 1;
            ss_f = 1;
            ss_g = 1;
            ss_dot = 1;
            wait(0.30);

    }

    else if(sev_ch == 'K'){
            ss_a = 1;
            ss_b = 0;
            ss_c = 0;
            ss_d = 1;
            ss_e = 0;
            ss_f = 0;
            ss_g = 0;
            ss_dot = 1;
            wait(0.30);

    }

    else if(sev_ch == 'L'){
            ss_a = 1;
            ss_b = 1;
            ss_c = 1;
            ss_d = 0;
            ss_e = 0;
            ss_f = 0;
            ss_g = 1;
            ss_dot = 1;
            wait(0.30);

    }


    else if(sev_ch == 'M'){
            ss_a = 1;
            ss_b = 1;
            ss_c = 1;
            ss_d = 1;
            ss_e = 1;
            ss_f = 1;
            ss_g = 1;
            ss_dot = 1;
            wait(0.30);

    }

    else if(sev_ch == 'N'){
```

```
                ss_a = 1;
                ss_b = 1;
                ss_c = 1;
                ss_d = 1;
                ss_e = 1;
                ss_f = 1;
                ss_g = 1;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == 'O'){
                ss_a = 0;
                ss_b = 0;
                ss_c = 0;
                ss_d = 0;
                ss_e =0 ;
                ss_f = 0;
                ss_g = 1;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == 'P'){
                ss_a = 0;
                ss_b = 0;
                ss_c = 1;
                ss_d = 1;
                ss_e = 0;
                ss_f = 0;
                ss_g = 0;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == 'Q'){
                ss_a = 0;
                ss_b = 0;
                ss_c = 0;
                ss_d = 0;
                ss_e = 0;
                ss_f = 0;
                ss_g = 0;
                ss_dot = 0;
                wait(0.30);

        }
```

```

    else if(sev_ch == 'R'){
            ss_a = 0;
            ss_b = 0;
            ss_c = 0;
            ss_d = 1;
            ss_e = 0;
            ss_f = 0;
            ss_g = 0;
            ss_dot = 1;
            wait(0.30);

    }

    else if(sev_ch == 'S'){
            ss_a = 0;
            ss_b = 1;
            ss_c = 0;
            ss_d = 0;
            ss_e = 1;
            ss_f = 0;
            ss_g = 0;
            ss_dot = 1;
            wait(0.30);

    }

    else if(sev_ch == 'T'){
            ss_a = 0;
            ss_b = 0;
            ss_c = 0;
            ss_d = 1;
            ss_e = 1;
            ss_f = 1;
            ss_g = 1;
            ss_dot = 1;
            wait(0.30);

    }

    else if(sev_ch == 'U'){
            ss_a = 1;
            ss_b = 0;
            ss_c = 0;
            ss_d = 0;
            ss_e = 0;
            ss_f = 0;
            ss_g = 1;
            ss_dot = 1;
            wait(0.30);
```

```

        }

        else if(sev_ch == 'V'){
                ss_a = 1;
                ss_b = 0;
                ss_c = 0;
                ss_d = 0;
                ss_e = 0;
                ss_f = 0;
                ss_g = 1;
                ss_dot = 1;
                wait(0.30);

        }
               else if(sev_ch == 'W'){
                ss_a = 1;
                ss_b = 0;
                ss_c = 0;
                ss_d = 0;
                ss_e = 0;
                ss_f = 0;
                ss_g = 1;
                ss_dot = 1;
                wait(0.30);

        }


        else if(sev_ch == 'X'){
                ss_a = 1;
                ss_b = 0;
                ss_c = 0;
                ss_d = 1;
                ss_e = 0;
                ss_f = 0;
                ss_g = 0;
                ss_dot = 1;
                wait(0.30);

        }

        else if(sev_ch == 'Y'){
                ss_a = 1;
                ss_b = 0;
                ss_c = 0;
                ss_d = 1;
                ss_e = 1;
                ss_f = 0;
                ss_g = 0;
```

```
            ss_dot = 1;
            wait(0.30);

    }

    else if(sev_ch == 'Z'){
            ss_a = 0;
            ss_b = 0;
            ss_c = 1;
            ss_d = 0;
            ss_e = 0;
            ss_f = 1;
            ss_g = 0;
            ss_dot = 1;
            wait(0.30);

    }

            ss_a = 1;
            ss_b = 1;
            ss_c = 1;
            ss_d = 1;
            ss_e = 1;
            ss_f = 1;
            ss_g = 1;
            ss_dot = 1;
}

// Function for determing the character after storing the dot/dash pattern received
inline void find_char()
{

    if(count_ch==0)
    {

        return;
    }

    else if(count_ch==1)    // If the no of characters received is 1
    {                       // the letter can be either E or T
        if(ch[0]==DOT)
        {
        pc.printf("E");     // Print the interpreted character on the terminal
        seven_seg_disp('E');
        }

        else if (ch[0]==DASH)
        {
        pc.printf("T");
        seven_seg_disp('T');
        }
    }

    else if(count_ch==2)
    {
        if((ch[0]==DOT)&&(ch[1]==DASH))
        {
            pc.printf("A");
        seven_seg_disp('A');
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT))
        {
            pc.printf("I");
        seven_seg_disp('I');
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT))
        {
            pc.printf("N");
        seven_seg_disp('N');
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH))
        {
            pc.printf("M");
        seven_seg_disp('M');
        }
    }

    else if(count_ch==3)
    {
        if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT))
        {
            pc.printf("S");
        seven_seg_disp('S');
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT)
            )
        {
            pc.printf("R");
        seven_seg_disp('R');
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
            ))
        {
            pc.printf("W");
        seven_seg_disp('W');
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
            )
        {
            pc.printf("D");
        seven_seg_disp('D');
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
            ))
        {
            pc.printf("G");
        seven_seg_disp('G');
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH
            ))
        {
            pc.printf("K");
        seven_seg_disp('K');
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
           DASH))
        {
            pc.printf("O");
        seven_seg_disp('O');
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
            )
        {
            pc.printf("U");
        seven_seg_disp('U');
        }


    }

    else if(count_ch==4)
    {
        if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch
            [3]==DOT))
        {
            pc.printf("H");
        seven_seg_disp('H');
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DOT)&&(ch[3]==DOT))
        {
            pc.printf("L");
           seven_seg_disp('L');
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)&&(ch[3]==DOT))
        {
            pc.printf("F");
            seven_seg_disp('F');
        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DASH))
        {
            pc.printf("V");
            seven_seg_disp('V');
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT)&&(ch[3]==DOT))
        {
            pc.printf("Z");
            seven_seg_disp('Z');
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH)&&(ch[3]==DOT))
        {
            pc.printf("P");
           seven_seg_disp('P');
        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH)&&(ch[3]==DASH))
        {
            pc.printf("J");
          seven_seg_disp('J');
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH)&&(ch[3]==DASH))
        {
            pc.printf("Y");
         seven_seg_disp('Y');
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DASH))
        {
            pc.printf("X");
         seven_seg_disp('X');
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT)&&(ch[3]==DASH))
        {
            pc.printf("Q");
         seven_seg_disp('Q');
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DASH)&&(ch[3]==DOT))
        {
            pc.printf("C");
         seven_seg_disp('C');
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DOT))
        {
            pc.printf("B");
         seven_seg_disp('B');
        }
    }

    else if(count_ch==5)
    {
        if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("5");
            seven_seg_disp('5');

        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)&&(ch[3]==DOT)&&(ch[4]==DASH))
        {
            pc.printf("4");
            seven_seg_disp('4');

        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DOT)
            &&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("3");
            seven_seg_disp('3');

        }

        else if((ch[0]==DOT)&&(ch[1]==DOT)&&(ch[2]==DASH)
            &&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("2");
            seven_seg_disp('2');

        }

        else if((ch[0]==DOT)&&(ch[1]==DASH)&&(ch[2]==DASH
            )&&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("1");
            seven_seg_disp('1');
        }

        else if((ch[0]==DASH)&&(ch[1]==DOT)&&(ch[2]==DOT)
            &&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("6");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==DOT
            )&&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("7");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
            DASH)&&(ch[3]==DOT)&&(ch[4]==DOT))
        {
            pc.printf("8");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
            DASH)&&(ch[3]==DASH)&&(ch[4]==DOT))
        {
            pc.printf("9");
        }

        else if((ch[0]==DASH)&&(ch[1]==DASH)&&(ch[2]==
            DASH)&&(ch[3]==DASH)&&(ch[4]==DASH))
        {
            pc.printf("0");
            seven_seg_disp('0');
        }

    }


}

// Counter for no of spaces
int count_space = 0;
bool bool_space = false;    // Flag for displaying space

// Function for handling space
inline void space()
{
    ss_a = 0;
    wait(0.05);
    ss_a = 1;

    count_space++;  // Increment the no of spaces received

    if(count_space<3){  // If there are two between letters then interpret the letter

    // depending on count, determine the character
    find_char();

    count_ch = 0;   // Reset the count and array
    ch[0] = 0;
    ch[1] = 0;
    ch[2] = 0;
    ch[3] = 0;
    ch[4] = 0;
    bool_space = false;
    }

    else if ((count_space>=5)&&(bool_space==false)) // Display space
    {
     bool_space = true;

    pc.printf("space");


    }

}

// Function for handling dot
```

```
inline void dot()
{
    count_space = 0;
    ss_d = 0;
    wait(0.05);
    ss_d = 1;

    buzz=1;
    wait(0.05);
    buzz = 0;
    if(count_ch>=5)
    {
    space();
    }
    else
    {
        ch[count_ch] = DOT;
        count_ch++;
    }
}

// Function for handling dash
inline void dash()
{
    count_space = 0;
    ss_g = 0;
    wait(0.05);
    ss_g = 1;

    buzz=1;
    wait(0.15);
    buzz = 0;

    if(count_ch>=5)
    {
    space();
    }
    else
    {
        ch[count_ch] = DASH;
        count_ch++;
    }}

// Function for turning off the 7 segment display
void segment_off()
{
    ss_a = 1;
    ss_b = 1;
    ss_c = 1;
    ss_d = 1;
    ss_e = 1;
    ss_f = 1;
    ss_g = 1;
    ss_dot = 1;

}

// Function for turning on the 7 segment display
void segment_on()
{
    ss_a = 0;
    ss_b = 0;
    ss_c = 0;
    ss_d = 0;
    ss_e = 0;
    ss_f = 0;
    ss_g = 0;
    ss_dot = 0;
}

// Debugging function
void key_rise_int1()
{
    wait(0.01);
    if(key)
    {
        pc.printf("pressed");
    }

}

// Debugging function
void key_fall_int1()
{
    wait(0.01);
    if(!key)
    {
        pc.printf("released");

    }
}

// Interrupt handler for key press
void key_rise_int()
{

    wait(0.01);
    //read the value of the key after 10ms and set the
       flag
    if(key)
    {
        timer_400.stop();   // Stop the space timer
        flag = true;
        //start a timer
        timer_key.start();  // Start the button press timer
       // pc.printf(”key pressed”);
    }

    else
    {
        flag = false;
    }

}

// Interrupt handler for key release
void key_fall_int()
{
    wait(0.01);

    if(!key)
    {
        //pc.printf(”key released”);
        //timer_400.reset();
        // stop the timer
        timer_key.stop();

        // read the timer value and decide if dot, dash
        int timer_key_val = timer_key.read_ms();

        if((timer_key_val>40)&&(timer_key_val<220)&&(flag==true) )
        {  // dot received
            /*myled1 = 1;
            wait(0.05);
            myled1 = 0;*/
            dot();
        }

        // dash received
        else if (timer_key_val>220){
            /*myled2 = 1;
            wait(0.05);
            myled2 = 0;*/
            dash();
        }

        // reset the timer
        timer_key.stop();
        timer_key.reset();
        flag=0;
        timer_400.reset();  // Reset and restart the space timer
        timer_400.start();

    }
}



int main() {


    // Check the leds
    myled1 = 0;
    myled2 = 1;
    myled3 = 1;
    myled4 = 1;

    wait(0.5);
    myled1 = 0;
    myled2 = 0;
    myled3 = 0;
    myled4 = 0;

    // Check the seven segment
    segment_off();
    wait(0.5);
    segment_on();
    wait(0.5);
    segment_off();

    pc.baud(9600);
    row = 1;


    // start the timer for 400ms
    timer_400.start();

    key.rise(&key_rise_int);    // Set up the interrup handler
    key.fall(&key_fall_int);


    while(1) {
        // if timer value hits 400 ms, then reset the timer and read as space.
        if(timer_400.read_ms()>=400)
        {
```

```
        timer_400.reset();

        if(!flag)
        {
        //pc.printf(”Got a space\n”);
        /*myled3 = 1;
        wait(0.05);
        myled3 = 0;*/

            space();

        }
        }
    }
}
```

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.