PHYSICS 1600: Introduction to Modern Technology

Fall 2020

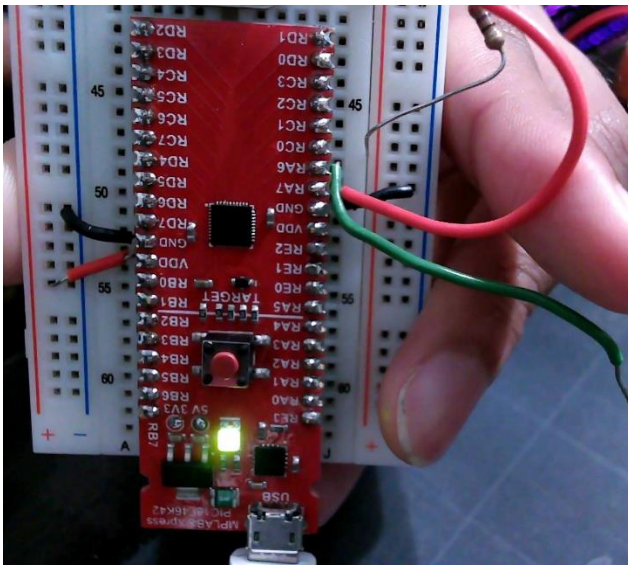
“Project Written Report”

Manmohan Badhesa

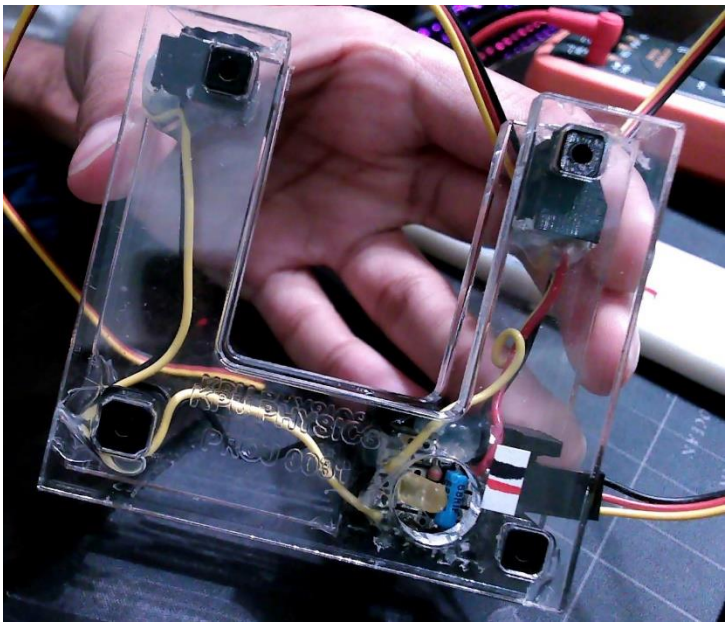
Introduction:

To program, using MPLAB software in C programming language, a PIC18F46K42 Microcontroller for a burglar alarm system. Project will need a PIC18F46K42 microcontroller, breadboard, keypad, photo gate, a speaker, and a bicolour LED. You will also need software plugins and PUTTY software. The photo gate is an infrared sensor that can detect human and other solid discrete object motion.

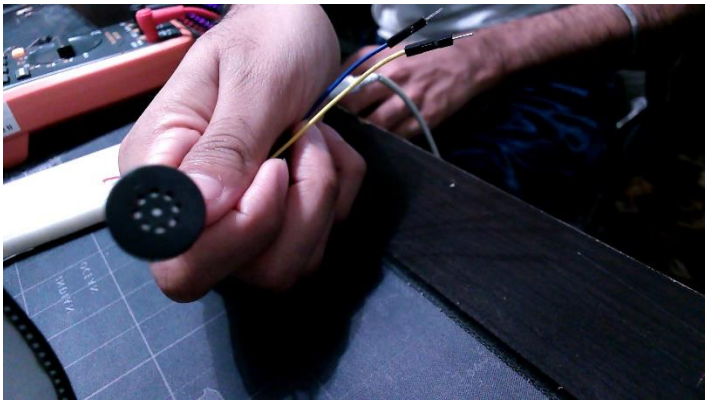
Equipment:



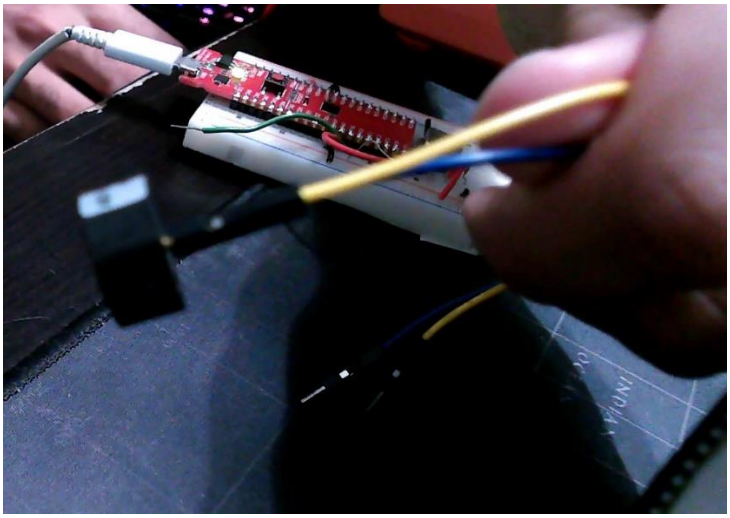
Picture 1. PIC18F46K42 Microcontroller on a breadboard



Picture 2. Photo gate used as motion sensor.



Picture 3. Speaker

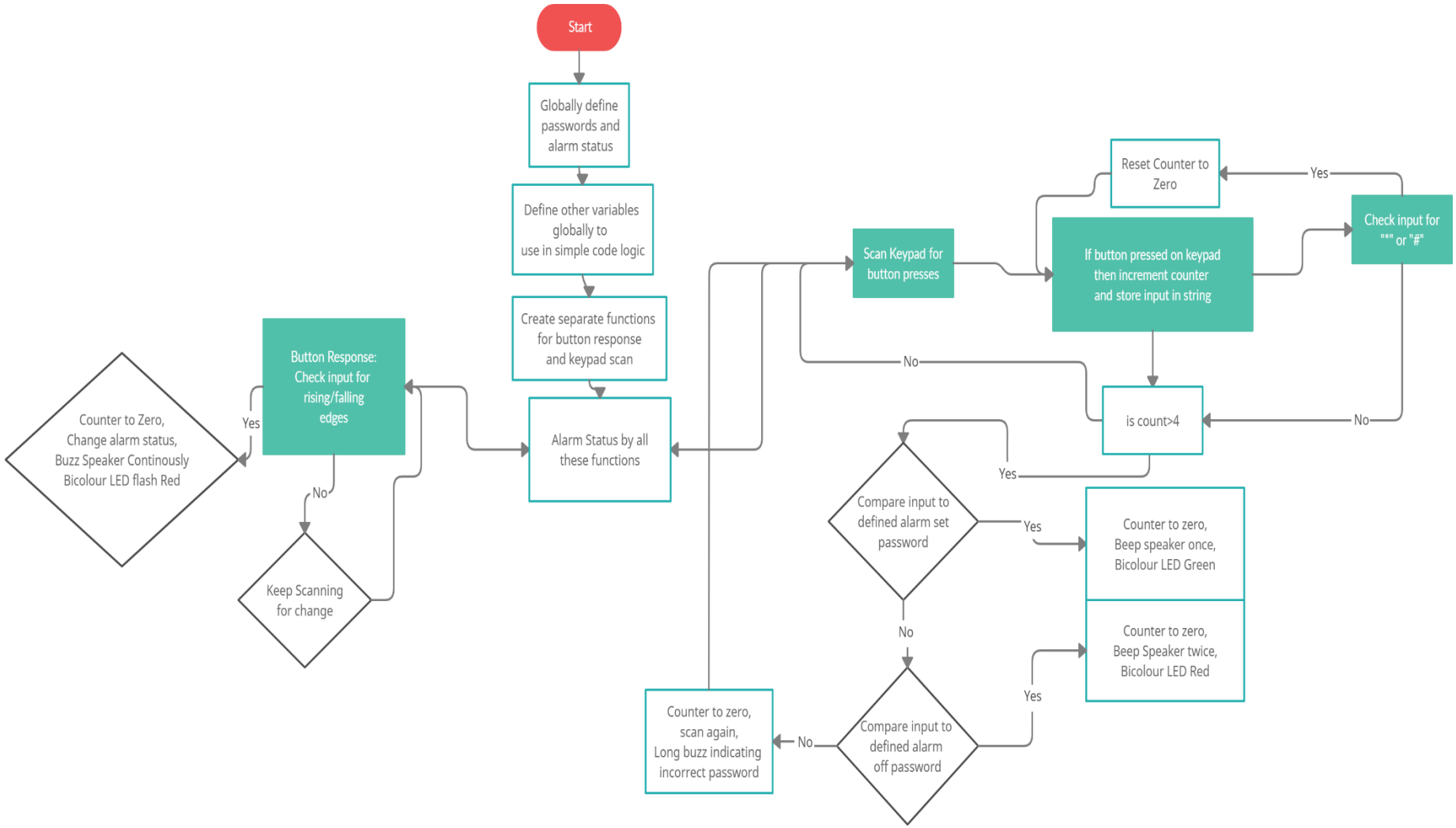


Picture 4. Speaker from the Side


Picture 5. Keypad with 7 pin connectors.

The code written provides a basic burglar alarm system that notifies developer when the system is activated, off, or triggered by an intruder. The code has separate functions that read in the input from either a keypad or photo gate. They both are button press inputs at different pins on the Microcontroller. The code then compares an input from the user to predefined passwords to turn on or turn off the alarm system.

A Pulse Width Modulation (PWM) was used for the speaker to make a sound. The function uses Timer2 to set the duty cycle of a signal. The duty cycle of a signal is the ratio of the signal being at a “on” level to the period of the signal. 50% duty cycle was used in this module, but the duty cycle value needed to be adjusted due to change of one parameter called PWMDC value (PR2) for PWM5. The new duty cycle value could be calculated by Duty Cycle % * 4.0 * (PR2+1)/100.0.  The percentage of the duty cycle is used at 50% to get the required frequency of the signal which is at half the duty cycle.


Start
Globally define passwords and alarm status
Define other variables globally to use in simple code logic
Create separate functions for button response and keypad scan
Alarm Status by all these functions
Button Response: Check input for rising/falling edges
Yes
Counter to Zero, Change alarm status, Buzz Speaker Continously Bicolour LED flash Red
No
Keep Scanning for change
Scan Keypad for button presses
If button pressed on keypad then increment counter and store input in string
Check input for "*" or "#"
Yes
Reset Counter to Zero
No
is count>4
No
Yes
Compare input to defined alarm set password
Yes
Counter to zero, Beep speaker once, Bicolour LED Green
No
Compare input to defined alarm off password
Yes
Counter to zero, Beep Speaker twice, Bicolour LED Red
No
Counter to zero, scan again, Long buzz indicating incorrect password

```c
//Bicolour LED at pins RD1 and RD2 with long leg at the pin you want to me make
//Connect pins of keypad in increasing order and group rows and columns separately
//to make it easier to remember as follows:
//Column pins RC1, RC2, RC3 in chronological order for convenience
//Row pins at RC4, RC5, RC6, RC7
//PWM5 at 50% using Timer2
//TMR2 using clock source FOSC/4, prescaler 1:128, and period of 4 ms
//Optional for Non-Blocking Delays:
//TMR0 16 bit, prescaler 1:32, FOSC/4 and 600 ms period


#include <stdio.h>    // C Library for printf()

#include <string.h>  // C Libary for strcmp function
#include "buttons.h" //contains function for detecting rising/falling edges for button presses
#include "putty.h"  // Library for clearPuTTY

#include "mcc_generated_files/mcc.h"

//Globally defined strings for passwords and string to store human attempt
char password_attempt[6];
char alarmset[]="1357*";
char alarmoff[]="1357#";
int alarm_status=0;  //integer used as a status to alternate between conditions

//Function for using photo gate as a button on PIC microcontroller
//has_switchN_changed used as global variable to detect button presses
void buttonResponse(void);
int has_switch1_changed=0,has_switch2_changed=0, has_switch3_changed=0, has_switch4_changed=0;
int has_switch_changed=0;

//Function for scanning the keypad for button presses
int scan_keypad();
unsigned int k;
int i=0;    //output from scan_keypad
int j=0;    //int used to start code when button pressed

//Code for adding beeps using Non-Blocking Delays
void beeps(void);
unsigned int q=55536;
```

```
40  unsigned int g=55536;
41
42  //Globally defined counter
43  int counter=0;
44
45  /*
46                          Main application
47   */
48  void main(void)
49  {
50      // Initialize the device
51      SYSTEM_Initialize();
52      //to check if human input matches either defined passwords alarmset or alarmoff
53      unsigned int compare,compare2;
54
55      clearPuTTY();
56
57      //Speaker should be quiet at the beginning of program which it isn't sometimes
58      PWM5_LoadDutyValue(0);
59
60      while (1)
61      {
62          // Add your application code
63
64          //Most of printf statements can be used by the developer to find errors in code
65          //and see what is actually outputting, so will keep them in code
66
67
68          buttonResponse();  //start looking for rising/falling edge at photo gate
69          k=scan_keypad();   //start keypad scan every cycle in the while loop
70          //printf("\n\r%u",j);
71          if(j == 1)          //only starts when button is pressed
72          {
73              //printf("\n\r counter %u",counter);
74
75              password_attempt[counter]=i;    //stores int value in a string each
76                                              //time button pressed on keypad
77
78
79              counter++;                      //increments counter every time button pressed
```

```
79              counter++;                      //increments counter every time button pressed
80              if(counter == 5) password_attempt[counter]= '\0';  //used to end string
81
82              if(counter>4 )                  //starts only when input as enough characters
83                  {
84                      counter=0;              //resets counter for successive attempts to continue
85                      //printf("\n\rInput characters: %u %u %u %u %u ",password_attempt[0],password_attempt[1],
86                                  //password_attempt[2],password_attempt[3],password_attempt[4],password_attempt[5]);
87                      //printf("\n\ralarm set %u %u %u %u %u",alarmset[0],alarmset[1],alarmset[2],alarmset[3],alarmset[4]);
88                      //printf("\n\ralarm off %u %u %u %u %u",alarmoff[0],alarmoff[1],alarmoff[2],alarmoff[3],alarmoff[4]);
89
90                      compare=strncmp(password_attempt,alarmset,5);       //compres string
91                                                                          //with predefined
92                                                                          //password alarmset
93                      //printf("\n\rcompare= %u",compare);
94
95                      if(compare == 0)                //starts when input matches password attempt
96                                                      //alarmset only
97                      {
98                          alarm_status=1;             //changes the alarm status to turn on alarm
99                          printf("\n\ralarm on %u",alarm_status);
100                         PR2=100;                    //used to create buzz sound of around 600 Hz
101                         PWM5_LoadDutyValue(202);    //to make the PWM5 at 50% for the correct frequency
102                         DELAY_milliseconds(500);    //runs sound for 500 milliseconds
103                         PWM5_LoadDutyValue(0);      //then turns off sound, resulting in single beep
104                     }
105
106
107                     compare2=strncmp(password_attempt,alarmoff,5);       //compres string
108                                                                         //with predefined
109                                                                         //password alarmoff
110                     //printf("\n\rcompare2= %u",compare2);
111                     //printf("\n\rpassword %s  %u ",password,strlen(password_attempt));
112
113                     if(compare2 == 0)                       //starts only if input matches with predefined
114                                                             //alarm off password
115                         {
116                             alarm_status=0;                 //changes, alarm status to off
117                             printf("\n\ralarm off %u",alarm_status);
118                             //below is similar to the code above to create buzz beep but two beeps this time
```

```
                        //below is similar to the code above to create buzz beep but two beeps this time
                        PR2=100;
                        PWM5_LoadDutyValue(202);
                        DELAY_milliseconds(180);
                        PWM5_LoadDutyValue(0);
                        DELAY_milliseconds(180);
                        PWM5_LoadDutyValue(202);
                        DELAY_milliseconds(180);
                        PWM5_LoadDutyValue(0);

                    }
                if(compare != 0 && compare2 != 0)       //only if the input password attempt does not
                                                        //match with either predefined passwords
                {
                    printf("\n\rwrong password");
                    //below is a longer buzz sound of 2 seconds to notify user of
                    //wrong password
                    PR2=100;
                    PWM5_LoadDutyValue(202);
                    DELAY_milliseconds(2000);
                    PWM5_LoadDutyValue(0);
                }
                }

        }

                if(i == 35 || i == 42)                  //if user inputs "*" or "#"
                                                        //the string coounter goes to zero
                {
                    counter=0;
                }

        //Switch acts as if statement with multiple conditions
        //Used for alarm status conditions
        switch(alarm_status)
        {
            case 0: PWM5_LoadDutyValue(0);      //alarm is off and Red LED on

                    red_RD2_SetHigh();
                    green_RD1_SetLow();
```

```
                    green_RD1_SetLow();

                    break;
                case 1: green_RD1_SetHigh();        //Alarm is on and Green LED on
                    red_RD2_SetLow();

                    break;
                case 2:                             //Alarm triggered and LED flashes Red
                    green_RD1_SetLow();
                    red_RD2_SetHigh();
                    DELAY_milliseconds(225);
                    red_RD2_SetLow();
                    green_RD1_SetLow();
                    DELAY_milliseconds(175);
                    break;
            }

        }
        }

    //function for photo gate trigger response same as button
    void buttonResponse(void)
    {
            has_switch_changed = poll_switch1_for_gate(gate_RA7_GetValue());
            if ( has_switch_changed == 2 && alarm_status == 1)
                //detects falling edge first and then rising edge
            {
                alarm_status=2;             //changes alarm status to triggered ie. flash Red
                PR2=100;  //buzz frequency continously until alarm deactivated or reset
                PWM5_LoadDutyValue(202);

                k=scan_keypad();
            }
    }


    //scans keypad for button presses
    //be sure to wire accordingly as mentioned at beginning of file
    //Need different has_switchN_changed for each button on keypad, added in Buttons.c file
    //where N is an int used to differ the statements
```

```
196  //where N is an int used to differ the statements
197  //view Buttons.c for more details
198  int scan_keypad()
199  {
200      j=0;
201      i=0;
202      //Cycles through powering each column with a very small delay in between
203      // each of has_switch_changed statements wait for change in voltage at the row pins
204      //Outputs i as a character to allow string comparison for passwords
205      //printf statements can be changed to increase privacy,
206      //did not alter printf for the developer to use
207          col_RC1_SetHigh();
208          col_RC2_SetLow();
209          col_RC3_SetLow();
210
211          has_switch1_changed = poll_switch1_for_edges(row1_RC4_GetValue());
212          //printf("\n\rcol 1 powered\n\r");
213          if ( has_switch1_changed == 1 )
214          {
215              printf("\n\rbutton 1");
216              i='1';
217              j=1;
218              //printf("i = %u",i);
219          }
220
221          has_switch2_changed = poll_switch2_for_edges(row2_RC5_GetValue());
222
223          if(has_switch2_changed == 1)
224          {
225              printf("\n\rbutton 4");
226              i='4';
227              j=1;
228          }
229
230          has_switch3_changed = poll_switch3_for_edges(row3_RC6_GetValue());
231
232          if(has_switch3_changed==1)
233          {
234              printf("\n\rbutton 7");
235              i='7';
```

```
            i='7';
            j=1;
        }

        has_switch4_changed = poll_switch4_for_edges(row4_RC7_GetValue());

        if(has_switch4_changed==1)
        {
            printf("\n\rbutton *");
            i='*';
            j=1;
        }


        DELAY_milliseconds(1);

        col_RC2_SetHigh();
        col_RC1_SetLow();
        col_RC3_SetLow();
        //printf("\n\rcol 2 powered\n\r");

        has_switch1_changed = poll_switch5_for_edges(row1_RC4_GetValue());
        //printf("\n\rcol 1 powered\n\r");
        if ( has_switch1_changed == 1 )
        {
            printf("\n\rbutton 2");
            i='2';
            j=1;
            //printf("i = %u",i);
        }

        has_switch2_changed = poll_switch6_for_edges(row2_RC5_GetValue());

        if(has_switch2_changed == 1)
        {
            printf("\n\rbutton 5");
            i='5';
            j=1;
        }

```

```
274
275             has_switch3_changed = poll_switch7_for_edges(row3_RC6_GetValue());
276
277             if(has_switch3_changed==1)
278             {
279                 printf("\n\rbutton 8");
280                 i='8';
281                 j=1;
282             }
283
284             has_switch4_changed = poll_switch8_for_edges(row4_RC7_GetValue());
285
286             if(has_switch4_changed==1)
287             {
288                 printf("\n\rbutton 0");
289                 i='0';
290                 j=1;
291             }
292
293
294             DELAY_milliseconds(1);
295
296             col_RC2_SetLow();
297             col_RC1_SetLow();
298             col_RC3_SetHigh();
299             //printf("\n\rcol 3 powered");
300
301             has_switch1_changed = poll_switch9_for_edges(row1_RC4_GetValue());
302             //printf("\n\rcol 1 powered\n\r");
303             if ( has_switch1_changed == 1 )
304             {
305                 printf("\n\rbutton 3");
306                 i='3';
307                 j=1;
308                 //printf("i = %u",i);
309             }
310
311             has_switch2_changed = poll_switch10_for_edges(row2_RC5_GetValue());
312
313             if(has_switch2_changed == 1)
```

```
313             if(has_switch2_changed == 1)
314             {
315                 printf("\n\rbutton 6");
316                 i='6';
317                 j=1;
318             }
319 
320             has_switch3_changed = poll_switch11_for_edges(row3_RC6_GetValue());
321 
322             if(has_switch3_changed==1)
323             {
324                 printf("\n\rbutton 9");
325                 i='9';
326                 j=1;
327             }
328 
329             has_switch4_changed = poll_switch12_for_edges(row4_RC7_GetValue());
330 
331             if(has_switch4_changed==1)
332             {
333                 printf("\n\rbutton #");
334                 i='#';
335                 j=1;
336 
337             }
338 
339             return i,j;
340     }
341 
342 
343     //Can use non-blocking delays but would need to do calculations for time period
344     void beeps(void)
345     {
346         PR2=100;
347         PWM5_LoadDutyValue(202);
348         TMR0IF = 0;       // clear flag
349         TMR0_WriteTimer(g);
350         while(!TMR0IF)
351         {
352             buttonResponse();
```

```
337             }
338
339             return i,j;
340     }
341
342
343     //Can use non-blocking delays but would need to do calculations for time period
344     void beeps(void)
345     {
346         PR2=100;
347         PWM5_LoadDutyValue(202);
348         TMR0IF = 0;       // clear flag
349         TMR0_WriteTimer(g);
350         while(!TMR0IF)
351         {
352             buttonResponse();
353         }
354
355         PWM5_LoadDutyValue(0);
356         TMR0IF = 0;       // clear flag
357         TMR0_WriteTimer(g);
358         while(!TMR0IF)
359         {
360             buttonResponse();
361         }
362
363         PWM5_LoadDutyValue(202);
364         TMR0IF = 0;       // clear flag
365         TMR0_WriteTimer(g);
366         while(!TMR0IF)
367         {
368             buttonResponse();
369         }
370
371         PWM5_LoadDutyValue(0);
372     }
373     /**
374      End of File
375     */
```