```c
//Kunal Mukherjee
//Project 1: Automatic Door
//2/16/2019
#include "stm32f446.h"
#include "stm32l476xx.h"
#include <stdio.h>
#include <stdlib.h>

/*stm446Template.c July 1, 2017
*/

//*Global functions
int getKeyPressed(void);
void delayOne(void);
void delayTwo(void);
void delayTen(void);

//*initialize global constants
int codeInputGottenFirst = 0; //set the 6-digit code
int codeInputGotten = 0;      //get the user input complete
int codeInputStatus = 0;      //flag to see if xode is correct or not

int code [6]; //the got set
int inputCode [6]; //the user input code

int flag1 = 0;
int flag3 = 0;
int flag5 = 0;
int selectIQ = 99;

int main()
 {
  //*Initialize varibales
  int i;
  int charPres = 99;
  int codeIndexFirst = 0;
  int codeIndex = 0;
  int codeCheck = 0;
  int reset = 0;

  //Enabling Clock bits
  RCC_AHB1ENR |= 1;     //Bit 0 is GPIOA clock enable bit
  RCC_AHB1ENR |= 4;     //Bit 3 is GPIOC clock enable bit

  //I/O bits
  //KeyPad
  GPIOC_MODER &= ~(3 << (2 * 1)); //C1 = input
  GPIOC_MODER &= ~(3 << (2 * 3)); //C3 = input
  GPIOC_MODER &= ~(3 << (2 * 5)); //C5 = input

  GPIOC_MODER |= (1 << (2 * 2)); //C2 = output
  GPIOC_MODER |= (1 << (2 * 4)); //C4 = output
  GPIOC_MODER |= (1 << (2 * 6)); //C6 = output
  GPIOC_MODER |= (1 << (2 * 7)); //C7 = output

  GPIOC_OTYPER |= (1 << (1 * 2)); //C2 = open-drain
  GPIOC_OTYPER |= (1 << (1 * 4)); //C4 = open-drain
  GPIOC_OTYPER |= (1 << (1 * 6)); //C6 = open-drain
  GPIOC_OTYPER |= (1 << (1 * 7)); //C7 = open-drain

  GPIOC_OSPEEDER |= (2 << (2 * 2)); //C2 = fast-speed
  GPIOC_OSPEEDER |= (2 << (2 * 4)); //C4 = fast-speed
  GPIOC_OSPEEDER |= (2 << (2 * 6)); //C6 = fast-speed
  GPIOC_OSPEEDER |= (2 << (2 * 7)); //C7 = fast-speed
  //GRIO_PUPDR = reset value is 0, so no pull-up/pull-down

  //LED and Solenoid
  GPIOA_MODER |= (1 << (2 * 0)); //A0 = output red
  GPIOA_MODER |= (1 << (2 * 1)); //A1 = output green
  GPIOA_MODER |= (1 << (2 * 6)); //A6 = output red
  GPIOA_MODER |= (1 << (2 * 7)); //A7 = output green
  GPIOA_MODER |= (1 << (2 * 5)); //A5 = output solenoid
```

```
73      //GPIOA_OTYPER default is push/pull-up/pull-down
74
75      GPIOA_OSPEEDER |= (2 << (2 * 0)); //A0 = fast-speed
76      GPIOA_OSPEEDER |= (2 << (2 * 1)); //A1 = fast-speed
77      GPIOA_OSPEEDER |= (2 << (2 * 6)); //A6 = fast-speed
78      GPIOA_OSPEEDER |= (2 << (2 * 7)); //A7 = fast-speed
79      GPIOA_OSPEEDER |= (2 << (2 * 5)); //A5 = fast-speed
80
81      //initialization of the code
82      for (i = 0; i < 6; i++){code[i] = 99; inputCode[i] = 99;}
83
84      //turn solenoid off
85      GPIOA_ODR |= (1 << (1 * 5)); //A5 = high- Solenoid
86
87      //Enable Interrupt
88      //Interrupt registers
89      NVIC_EnableIRQ(EXTI1_IRQn);//PC_1
90      NVIC_EnableIRQ(EXTI2_IRQn);//PC_3
91      NVIC_EnableIRQ(EXTI3_IRQn);//PC_5
92
93      //Connect External Line to the GPI
94      RCC_APB2ENR |= RCC_APB2ENR_SYSCFGEN;
95
96      SYSCFG->EXTICR[0] &= ~SYSCFG_EXTICR1_EXTI1;
97      SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI1_PC;
98
99      SYSCFG->EXTICR[1] &= ~SYSCFG_EXTICR1_EXTI3;
100     SYSCFG->EXTICR[1] |= SYSCFG_EXTICR1_EXTI3_PC;
101
102     SYSCFG->EXTICR[2] &= ~SYSCFG_EXTICR2_EXTI5;
103     SYSCFG->EXTICR[2] |= SYSCFG_EXTICR2_EXTI5_PC;
104
105     //Rising trigger selection
106     //0 = trigger disabled, 1 = trigger enabled
107     EXTI->RTSR1 |= EXTI_RTSR1_RT1;
108     EXTI->RTSR1 |= EXTI_RTSR1_RT3;
109     EXTI->RTSR1 |= EXTI_RTSR1_RT5;
110
111     //Interrupt Mask Register
112     //0 = marked, 1 = not masked (enabled)
113     EXTI->IMR1 |= EXTI_IMR1_IM1;
114     EXTI->IMR1 |= EXTI_IMR1_IM3;
115     EXTI->IMR1 |= EXTI_IMR1_IM5;
116
117      //Main program loop
118      while(1)
119      {
120         while (!codeInputGottenFirst) //getting the inout code for the first time
121         {
122           delayOne();
123           if(flag1 || flag3 || flag3)
124           {
125             charPres = selectIQ;
126           }
127
128           if (charPres != 99)
129           {
130             code[codeIndexFirst] = charPres;
131             codeIndexFirst++;
132             GPIOA_ODR |= (1 << (1 * 0)); //A0 = high; //red
133             delayTwo();
134             GPIOA_ODR &= ~(1 << (1 * 0)); //A0 = low; //red
135           }
136
137           if (codeIndexFirst == 6){codeInputGottenFirst = 1;}
138         }
139         delayTwo();
140         GPIOA_ODR |= (1 << (1 * 0)); //A0 = high; //red;
141
142         while (!codeInputGotten) //getting the inout code
143         {
144           delayOne();
```

```
145
146           if(flag1 || flag3 || flag3)
147           {
148             charPres = selectIQ;
149           }
150
151           if (charPres != 99)
152           {
153             inputCode[codeIndex] = charPres;
154             codeIndex++;
155             GPIOA_ODR |= (1 << (1 * 1)); //A1 = high;
156             delayTwo();
157             GPIOA_ODR &= ~(1 << (1 * 1)); //A1 = low;
158           }
159
160           if (codeIndex == 6){codeInputGotten = 1;}
161         }
162
163         //check to see reset
164         for (i = 0; i < 6; i++)
165         {
166           if (inputCode[i] != 0)
167           {
168             reset = 0;
169             break;
170           }else{
171             reset = 1;
172           }
173         }
174
175         if (reset == 0)
176         {
177             //check to see if code is same
178             for (i = 0; i < 6; i++)
179             {
180               if (code[i] == inputCode[i])
181               {
182                 codeInputStatus = 1;
183               }
184               else
185               {
186                 codeInputStatus = 0;
187                 break;
188               }
189             }
190
191             if (codeInputStatus == 1) //code is right
192             {
193               delayTwo();
194               GPIOA_ODR &= ~(1 << (1 * 5)); //A5 = low- Solenoid ON
195               for (i = 0; i < 5; i++)
196               {
197                 GPIOA_ODR |= (1 << (1 * 6)); //A6 = high;
198                 delayTwo();
199                 GPIOA_ODR &= ~(1 << (1 * 6)); //A6 = low;
200                 delayTwo();
201               }
202               delayTen();
203               GPIOA_ODR |= (1 << (1 * 5)); //A5 = high- Solenoid OFF
204
205             }
206             else
207             {
208               delayTwo();
209               for (i = 0; i < 5; i++)
210               {
211                 GPIOA_ODR |= (1 << (1 * 7)); //A7 = red
212                 delayTwo();
213                 GPIOA_ODR &= ~(1 << (1 * 7)); //A7 = red
214                 delayTwo();
215               }
216             }
```

```
217         }else{
218             delayTwo();
219             GPIOA_ODR &= ~(1 << (1 * 0)); //A0 = high; //red;
220             codeInputGottenFirst = 0;
221             codeIndexFirst = 0;
222             for (i = 0; i < 6; i++){code[i] = 99;}
223         }
224 
225 
226         //clear the code entered
227         //initialization of the code
228         for (i = 0; i < 6; i++){inputCode[i] = 99;}
229         codeIndex = 0;
230         codeInputStatus = 0;
231         codeInputGotten = 0;
232         delayOne();
233 
234       }
235 
236     }
237      void EXTI1_IRQHandler(void) //C1 = 2
238     {
239        //check for EXIT 1 interrupt flag
240       if ((EXTI->PR1 & EXTI_PR1_PIF1) == EXTI_PR1_PIF1)
241        {
242         if(! (GPIOC_IDR & (1 << 2))) //check if C2 is low, 3
243         {
244           selectIQ = 2;
245         }
246         if(! (GPIOC_IDR & (1 << 4))) //check if C4 is low, 5
247         {
248           selectIQ = 0;
249         }
250         if(! (GPIOC_IDR & (1 << 6))) //check if C6 is low, 7
251         {
252           selectIQ = 8;
253         }
254         if(! (GPIOC_IDR & (1 << 7))) //check if C7 is low, 8
255         {
256           selectIQ = 5;
257         }
258 
259         flag1 = 1;
260         //clear interrupt pending request
261         EXTI->PR1 |= EXTI_PR1_PIF1;
262        }
263     }
264 
265     void EXTI3_IRQHandler(void) //C3 = 4
266     {
267        //check for EXIT 3 interrupt flag
268       if ((EXTI->PR1 & EXTI_PR1_PIF3) == EXTI_PR1_PIF3)
269        {
270         if(! (GPIOC_IDR & (1 << 2))) //check if C2 is low, 3
271         {
272           selectIQ = 1;
273         }
274         if(! (GPIOC_IDR & (1 << 4))) //check if C4 is low, 5
275         {
276           selectIQ = 11;
277         }
278         if(! (GPIOC_IDR & (1 << 6))) //check if C6 is low, 7
279         {
280           selectIQ = 7;
281         }
282         if(! (GPIOC_IDR & (1 << 7))) //check if C7 is low, 8
283         {
284           selectIQ = 4;
285         }
286 
287         flag3 = 1;
288         //clear interrupt pending request
```

```
289         EXTI->PR1 |= EXTI_PR1_PIF3;
290       }
291     }
292
293      void EXTI5_IRQHandler(void) //C5 = 6
294     {
295        //check for EXIT 5 interrupt flag
296       if ((EXTI->PR1 & EXTI_PR1_PIF5) == EXTI_PR1_PIF5)
297        {
298         if(! (GPIOC_IDR & (1 << 2))) //check if C2 is low, 3
299         {
300           selectIQ = 3;
301         }
302         if(! (GPIOC_IDR & (1 << 4))) //check if C4 is low, 5
303         {
304           selectIQ = 10;
305         }
306         if(! (GPIOC_IDR & (1 << 6))) //check if C6 is low, 7
307         {
308           selectIQ = 9;
309         }
310         if(! (GPIOC_IDR & (1 << 7))) //check if C7 is low, 8
311         {
312           selectIQ = 6;
313         }
314
315         flag5 = 1;
316         //clear interrupt pending request
317         EXTI->PR1 |= EXTI_PR1_PIF5;
318       }
319     }
320
321
322     void delayOne(void)
323     {
324       int i,j;
325       for (i = 0; i < 5000; i++);
326       //for (i = 0; i < 16000; i++){ for (j = 0; j < 250; j++);} //1 sec
327     }
328
329     void delayTwo(void)
330     {
331       int i,j;
332       for (i = 0; i < 16000; i++){ for (j = 0; j < 350; j++);} //1 sec
333     }
334
335     void delayTen(void)
336     {
337      int i,j;
338      for (i = 0; i < 16000; i++){ for (j = 0; j < 1250; j++);} //10 sec
339     }
340
341
342
343
```