```
1   //Kunal Mukherjee;
2   //2/10/2019
3   //Proj 1
4
5   //the include files
6   #include "stm32l432.h"
7
8   //ADC that takes input with a channel
9   void ADC_Init(int channel);
10  void GPIO_Init(void);
11  void ADC_CLK_GPIO(void);
12
13  #define DELTA 130
14  #define MAX 5000 //2.5 ms = 5000/40000 * 20
15  #define MIN 1000// 0.5 ms = 1000/40000 * 20
16  #define MIDDLE 3000 //1.5 ms = 3000/40000 * 20
17
18  #define RIGHT_EMPTY 680 //at room light this is what the RIGHT_LED read
19                          //anything greater means that an IR has replected back
20                          // so it is seeing something
21  #define LEFT_EMPTY 668  //
22
23  #define DELAY 250 // wait for that may counts before resuming work
24
25  int main()
26  {
27
28    //assign the variables
29    unsigned int i, j, tmr2, choice, diff;
30    int right_sensor, left_sensor;
31
32    GPIO_Init();
33    ADC_CLK_GPIO();
34
35    //initilization of the variables
36    tmr2 = MIDDLE;
37    right_sensor = left_sensor = 0;
38
39    while(1)
40    {
41      ADC_Init(8); //setup adc of channel 8
42      ADC_CR |= (1 << 2); //start adc regular conversion
43      while((ADC_ISR & (1 << 2)) == 0); // end of regular sequence flag
44      left_sensor = ADC_DR & 0xFFF; //only look at 12 bit
45      ADC_CR |= (1 << 1); //turn ADC off
46
47      for (i = 0; i < DELAY; i++); //delay
48
49      ADC_Init(10); //setup adc for channel 10
50      ADC_CR |= (1 << 2); //start adc regular conversion
51      while((ADC_ISR & (1 << 2)) == 0); // end of regular sequence flag
52      right_sensor = ADC_DR & 0xFFF; //only look at 12 bit
53      ADC_CR |= (1 << 1); //turn ADC off
54
55      for (i = 0; i < DELAY; i++); //delay
56
57      //getting the diff betwen the sensors
58      if (right_sensor > left_sensor)
59      {
60          diff = right_sensor - left_sensor;
61      }else{
62          diff = left_sensor - right_sensor;
63      }
64
65      //nothing to follow
66      //so the finding algorithm
67      if (((right_sensor <= RIGHT_EMPTY)) &&
68          ((left_sensor <= LEFT_EMPTY) ))
69      {
70        if (choice == 0)
71        {
72          tmr2 += DELTA;
```

```
73            if (tmr2 > MAX)
74            { choice = 1; tmr2 = MAX;}
75          }
76          else
77          {
78            tmr2 -= DELTA;
79            if (tmr2 < MIN)
80            {choice = 0; tmr2 = MIN;}
81          }
82        }
83
84        //sensors sensed something
85        else
86        {
87           if (diff < 70) //the object is in the middle
88           {
89              //do nothing, item found
90           }
91
92          else {
93              if (right_sensor > left_sensor) //right sensor has sensed something
94              {
95                tmr2 += DELTA; //add some movement to right
96
97                if (tmr2 > MAX) //if max then do not move from the right
98                { tmr2 = MAX; }
99              }else{ //left sensor has sensed something
100               tmr2 -= DELTA;  // add some movement to the left
101
102               if (tmr2 < MIN) //if mov then do not move from the left
103               { tmr2 = MIN; }
104
105             }
106         }
107       }
108
109       //code to check if -180+180 motion is being read
110       /*if (choice == 0)
111         {
112           tmr2 += DELTA;
113           if (tmr2 > MAX)
114             choice = 1;
115         }else{
116           tmr2 -= DELTA;
117           if (tmr2 < MIN)
118             choice = 0;
119         }*/
120
121       //enter the new high time value to the CCR2 reg
122       TIM2_CCR2 = tmr2; //scale the value
123
124       for (i = 0; i < 10000; i++){for (j = 0; j < 5; j++);}; //delay
125     }
126    }
127
128   void GPIO_Init(void)
129   {
130     //clock initilaizations
131     RCC_AHB2ENR  |= (1 << 0);  //set the GPIOA clk
132     RCC_APB1ENR1 |= (1 << 0); //TIM2 en from APB1 peri clk enb reg
133
134     //GRIO setup
135     GPIOA_MODER   &= ~(3 << (2 * 1)); //clear the GPIOA mode bits
136     GPIOA_MODER   |=  (2 << (2 * 1)); //set port a1 is alternate 10
137     GPIOA_OTYPER  &= ~(1 << (1 * 1)); //open drain for a1
138     GPIOA_OTYPER  |=  (1 << (1 * 1)); //open drain for a1
139     GPIOA_OSPEEDR &= ~(3 << (2 * 1)); //high speed output
140     GPIOA_OSPEEDR |=  (2 << (2 * 1)); //high speed output
141     GPIOA_AFRL    |=  (1 << (4 * 1)); //alt func 1, port pin 5,
142                                       //control bit are 4 bit wide
143
144     TIM2_CR1 |= (1 << 7); //ARPE: Auto-reload preload enable
```

```
145     TIM2_PSC = 1;          //PSC set to 2 = 1 + 1
146     TIM2_ARR = 40000;      //50 Hz = 20  = ms; 4MHz/2 = 2MHz; 2MHz/40000 = 50Hz
147     TIM2_CCMR1 |= 0x6800; //Channel 2;
148                            //bit 11: OC2PE: Output compare 2 preload enable
149                            //0110: PWM mode 1 - In upcounting,
150                            //channel 1 is active as long as
151                            //TIMx_CNT<TIMx_CCR1else inactive.
152     TIM2_CCER |= (1 << 4);//CC1E: Capture/Compare 2 output enable.
153     TIM2_CCR2 |= MIDDLE;  //CCR2 is the value to be loaded in the actual
154                            //capture/compare 2 register (preload value).
155     TIM2_EGR |= (1 << 0); //UG: update event
156     TIM2_CR1 |= (1 << 0); //CEN: counter enabled
157   }
158
159   void ADC_CLK_GPIO(void)
160   {
161     //set up the adc clock
162     RCC_AHB2ENR  |= (1 << 13); //set ADC clk
163
164     //adc GPIO Setup
165     //mode default to analog for PA3 & PA5
166     GPIOA_PUPDR &= ~(3  << (2 * 3)); //PA3 to no pull or down
167     GPIOA_PUPDR &= ~(3  << (2 * 5)); //PA5 to no pull or down
168   }
169
170
171   void ADC_Init(int channel)
172   {
173     ADC_CR &= ~(1 << 0); //disable ADC
174
175     int i;  //a counter for .5 us
176
177     ADC_CR &= ~(1 << 29); //deep power mode cleared
178     ADC_CR |= (1 << 28); //set voltage reg
179
180     for (i=0; i <10000; i++);//wait for .5us
181
182     ADC_CCR |= (1 << 22); //VREF ENAB
183     ADC_CCR |= (1 << 16); //HCLK/1 (Synchronous clock mode) enb
184
185     ADC_ISR |= (1 << 0); //ADC ready
186     ADC_CR |= (1 << 0); //ENB ADC
187
188     while ((ADC_ISR & (1 << 0)) == 0); //wait till ADC is ready
189
190     ADC_SQR1 &= ~(31 << 6);
191     ADC_SQR1 |=  (channel << 6);//CH8 for A3 or CH10 for A5
192
193     ADC_CFGR |= (1 << 16); //DISCEN: Discontinuous mode for regular channels
194   }
195
```