# INFO-2055: Embedded systems project

## Customer Counter
## Sensors and actuators validation


| Crucifix Arnaud | Goffart Maxime | Joris Olivier |
|---|---|---|
| 170962 | 180521 | 182113 |


Academic year 2020 - 2021

# 1 Project idea (reminder)

We are going to use a microcontroller to manage the maximum number of customers allowed inside a shop in this period of health crisis.

Indeed, it will keep track of the number of persons allowed inside the shop. We will use an infrared sensor system to detect customers entering and leaving the shop. In order to limit the customers' flow, we will use 7 segments to display the number of customers that can still enter.

When powering up the system or after a reset, we will also use these 7 segments displays to configure the maximum number of customers that can be in the shop at the same time using buttons.

# 2 Hardware (updated)

We are using one infrared emitter for the entry and another one for the exit. Moreover, we are using a "central" circuit that will contain both receivers. Thus, we have 2 different kinds of circuits.

Thanks to this choice, we can choose how far we want each emitter to be from the receiving circuit as long as we stay in the limit of the emitters.

## 2.1 Schematic

The main modifications, compared to the previous schematic, are splitting the emitting and receiving parts, switching to 9V battery as a power source instead of 5V DC power supply using a jack connector, and using phototransistors rather than IR receivers.

### 2.1.1 Receiver

The receiving circuit contains both infrared receivers, both 7 segments, and buttons to set the limit of customers. Moreover, the LED D1 will be blinking while the manager chooses the maximum number of customers and staying switched on when the system is operational. See Figure 1 for the schematic.

### 2.1.2 Emitter

Each emitting circuit contains an infrared emitter with secondary components in order to regulate the voltage. See Figure 2 for the schematic.

## 2.2 Physical circuit

We will setup the system in the following way:

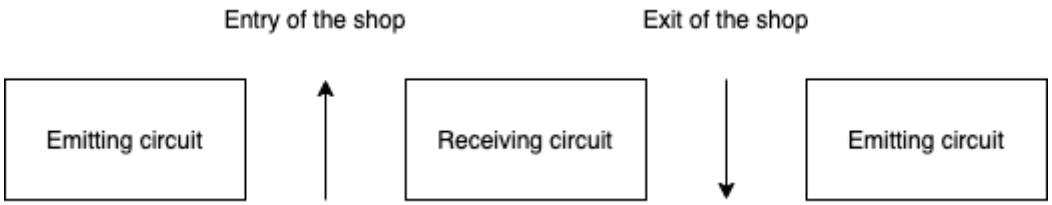


Figure 3: Abstract representation of the setup of the 3 circuits

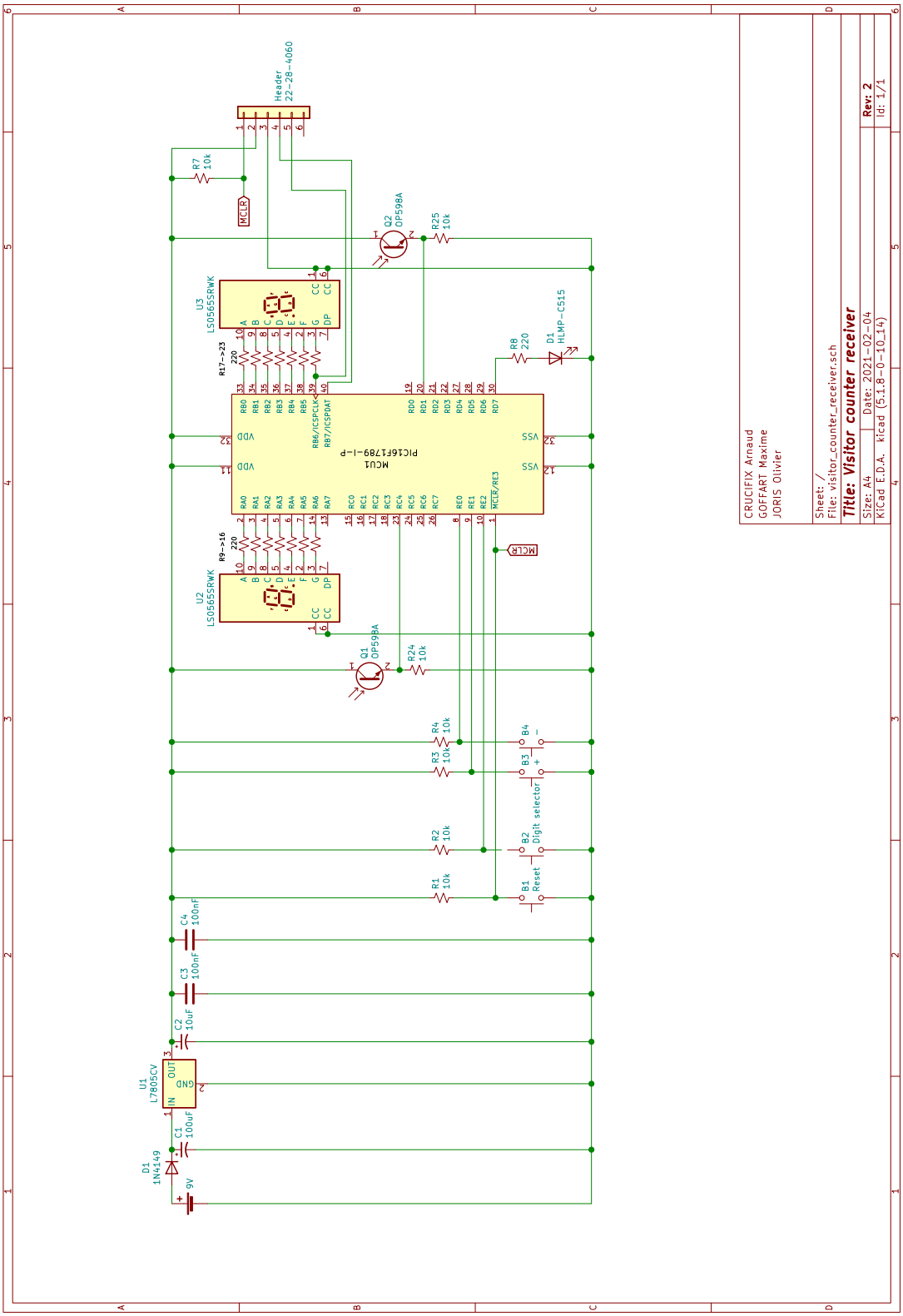


Figure 1: Schematic of the receiving circuit

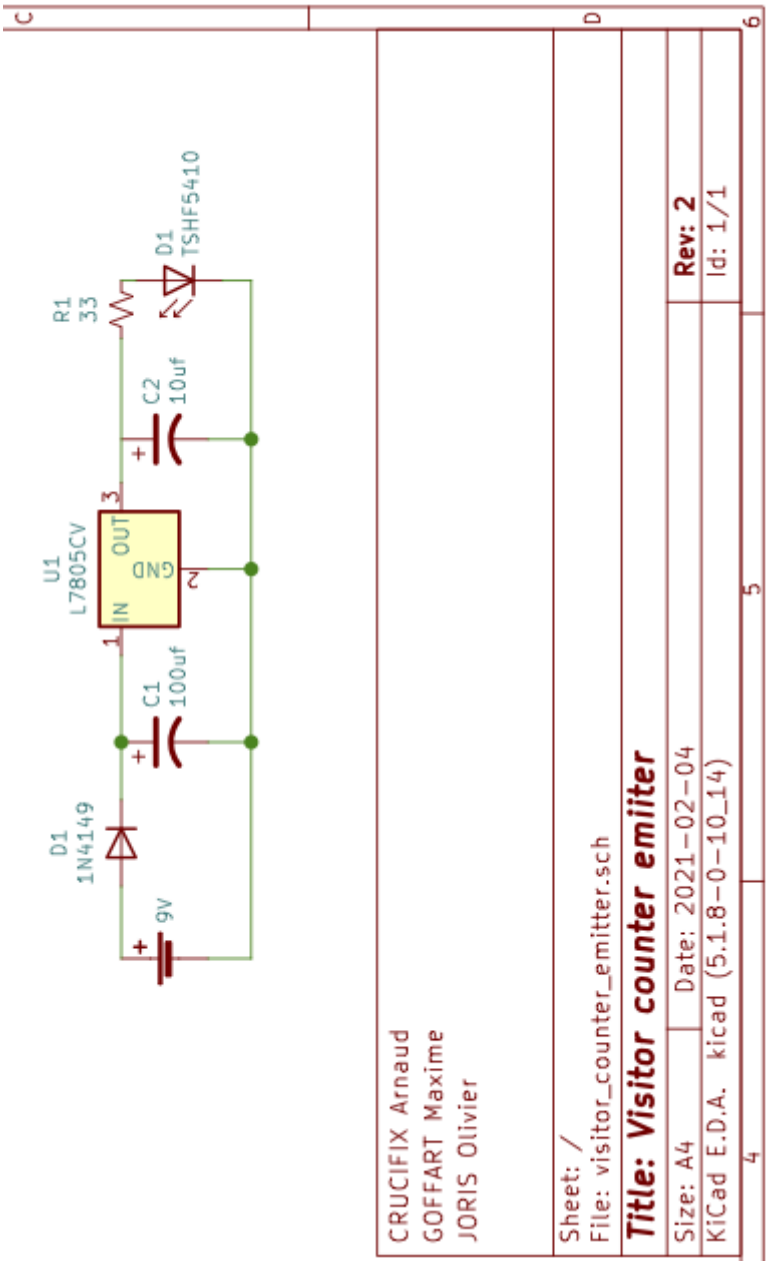


Figure 2: Schematic of the emitting circuit

The 3 circuits will have to be at a certain height in order to improve the detection. A trial-and-error method must be used in order to determine the optimal height.

# 3 Sensors

We decided to test our two kinds of sensors with separate codes for each one in order to be sure they are correctly operated in our circuit.

## 3.1 Phototransistors

To test our phototransistors, we decided to implement the following test. If someone (or just an object) is detected by the sensor, it lights up the green LED of the receiver (D1 on the schematic).

We implemented this test using this code :

```
    processor 16f1789
    #include "config.inc"

    PSECT text, abs, class=CODE, delta=2

    org  00h
    goto start

start:
    call initialisation
    goto main_loop

initialisation:
    movlb 01h            ;Go to bank 1
    clrf TRISC
    movlw 00010000B      ;Set the port pin types of RC
    movwf TRISC          ;RC4 is input

    clrf TRISD           ;All pins of PORTD are output
    movlw 00000010B      ;RD1 is input. Others are output.
    movwf TRISD          ;RD1 is input

    movlb 02h
    clrf LATD            ;Clear output of all PORTD

    movlb 07h            ;Go to bank 7
    movlw 00010000B
    movwf INLVLC         ;Set up schmitt trigger with CMOS levels for RC4

    movlw 00000010B
    movwf INLVLD         ;Set up schmitt trigger with CMOS levels for RD1

    movlb 03h
    movlw 00000000B
    movwf ANSELD         ;Set all PORTD as digital
    movlw 00000000B
    movwf ANSELC         ;Set all PORTC as digital

    ;configure the clock - 4MHz
    movlb 01h
    movlw 01101110B
    movwf OSCCON
    movlw 00000000B
```

```
44      movwf OSCTUNE
45
46      return
47
48  main_loop:
49      movlb 00h
50      btfss PORTC, 4              ;Read RC4 pin
51      goto no_pass_detected       ;If RC4 is '0', then go to no_pass_detected
52      goto pass_detected          ;Else, go to pass_detected
53      movlb 00h
54      btfss PORTD, 1              ;Read RD1 pin
55      goto no_pass_detected       ;If RD1 is '0', then go to no_pass_detected
56      goto pass_detected          ;Else, go to pass_detected
57
58  pass_detected:
59      movlb 02h
60      movlw 10000000B             ;Set RD7 output
61      movwf LATD
62      goto main_loop              ;Go back to the main loop
63
64  no_pass_detected:
65      movlb 02h
66      movlw 00000000B             ;Clear RD7 output
67      movwf LATD
68      goto main_loop              ;Go back to the main loop
```

Code 1: Phototransistors test code

### 3.2 Buttons

To test one button, we decided to implement the following test. If someone press the button (in this case, the one connected to RE0), it lights up the green LED of the receiver (D1 on the schematic).

We implemented this test using this code :

```
1       processor 16f1789
2       #include "config.inc"
3
4       PSECT text, abs, class=CODE, delta=2
5
6       org  00h
7       goto start
8
9   start:
10      call initialisation
11      goto main_loop
12
13  initialisation:
14      movlb 01h           ;Go to bank 1
15      movlw 00000001B     ;Set the port pin types of PORTE
16      movwf TRISE         ;RE0 is input
17
18      movlw 01111111B     ;Set the port pin types of PORTD
19      movwf TRISD         ;RD7 is output
20
21      movlb 02h
22      clrf LATD           ;Clear output of all PORTD
23
24      movlb 03h
```

```
25      movlw 00000000B
26      movwf ANSELE        ;Set all PORTE as digital
27      movlw 00000000B
28      movwf ANSELD        ;Set all PORTD as digital
29
30      ;configure the clock - 4MHz
31      movlb 01h
32      movlw 01101110B
33      movwf OSCCON
34      movlw 00000000B
35      movwf OSCTUNE
36
37      return
38
39  main_loop:
40      movlb 00h
41      btfss PORTE, 0      ;Read RE0 pin
42      goto button_is_off  ;If RE0 is '0', then go to button_is_off
43      goto button_is_on   ;Else go to button_is_on
44      goto main_loop
45
46  button_is_on:
47      movlb 02h
48      movlw 10000000B     ;Set RD7 output
49      movwf LATD
50      goto main_loop      ;Go back to the main loop
51
52  button_is_off:
53      movlb 02h
54      movlw 00000000B     ;Clear RD7 output
55      movwf LATD
56      goto main_loop      ;Go back to the main loop
```

Code 2: Buttons test code

# 4 Actuators

We decided to test our actuators, which are the 2 7-segments, independently of the rest of the system to be sure that the microcontroller can successfully interact with them without the potential interactions with other components.

## 4.1 7 segments

We are displaying numbers starting from 00 up to 99 by displaying the same digit on both the 7 segments. To do so, we are using the following code:

```
1       processor 16f1789
2       #include "config.inc"
3
4       PSECT text, abs, class=CODE, delta=2
5
6       org  00h
7       goto start
8
9       org 04h
10      goto interrupt_routine
11
12
```

```
13 array:
14     movf counter, 0 ;w <- counter (p.117 in datasheet)
15     addwf PCL, 1    ;w is added to PCL in order to jump to the corresponding
16                     ;line which is returning a literal value into w (retlw) and
17                     ;comes back to the originated call.
18     retlw 00111111B ;representation of 0 in the 7-segment(3F)
19     retlw 00000110B ;representation of 1 in the 7-segment(06)
20     retlw 01011011B ;representation of 2 in the 7-segment(5B)
21     retlw 01001111B ;representation of 3 in the 7-segment(4F)
22     retlw 01100110B ;representation of 4 in the 7-segment(66)
23     retlw 01101101B ;representation of 5 in the 7-segment(6D)
24     retlw 01111100B ;representation of 6 in the 7-segment(7D)
25     retlw 00000111B ;representation of 7 in the 7-segment(07)
26     retlw 01111111B ;representation of 8 in the 7-segment(7F)
27     retlw 01100111B ;representation of 9 in the 7-segment(6F)
28
29 start:
30     call initialisation
31     goto main_loop
32
33 initialisation:
34     movlb 01h           ;Go to bank 1
35
36     ;7-segment on the left side(U2)
37     clrf TRISA          ;All pins of PORTA = outputs
38
39     ;7-segment on the right side(U3)
40     clrf TRISB          ;All pins of PORTB = outputs
41
42     movlb 02h
43     clrf LATA           ;All outputs of PORTA = 0
44     clrf LATB           ;All outputs of PORTB = 0
45
46     ;Clock configuration - 4MHz - Internal oscillator
47     movlb 01h
48     movlw 01101110B
49     movwf OSCCON
50     movlw 00000000B
51     movwf OSCTUNE
52
53     ;Clear before enabling interrupts
54     movlb 00h
55     clrf TMR1H
56     clrf TMR1L
57     bcf PIR1, 0         ;TMR1IF = 0
58
59     ;Timer1 ON using a 1:8 prescaler
60     movlb 00h
61     movlw 00110001B     ;(p.217 in datasheet)
62     movwf T1CON         ;configure Timer1
63
64     ;Timer1 Interrupt ON
65     movlb 01h
66     movlw 00000001B
67     movwf PIE1          ;Enable timer 1 overflow interrupt
68     movlw 11000000B
69     movwf INTCON        ;Enable interrupt
70
71     ;Trigger an interrupt
72     movlb 00h
```

```
73      movlw 00001011B     ;the 8 most significant bits
74      movwf TMR1H
75      movlw 11011011B     ;the 8 least significant bits
76      movwf TMR1L
77
78      counter EQU 20h     ;address 20=General Purpose Register(p.33 in datasheet)
79      maxcounter EQU 21h  ;address 21=General Purpose Register(p.33 in datasheet)
80
81      ;set counter and maxcounter
82      movlb 00h
83      movlw 00000000B
84      movwf counter
85      movlw 00001010B
86      movwf maxcounter
87
88      return
89
90  main_loop:
91      goto main_loop
92
93  interrupt_routine:
94      movlb 00h
95      btfss PIR1, 0       ;timer1 overflow interrupt flag bit TMR1IF
96      RETFIE
97
98  timer_interrupt:
99      movlb 00h
100     bcf T1CON, 0        ;set the first bit to 0
101     movlw 00001011B
102     movwf TMR1H         ;reset register
103     movlw 11011011B
104     movwf TMR1L         ;reset register
105
106 increment_on:
107     call array          ;fetch the number in w based on the counter index
108     movlb 00h
109     movwf PORTA         ;pins of the first 7-segment high
110                         ;based on the number fetched
111     movwf PORTB         ;pins of the second 7-segment high
112                         ;based on the number fetched
113     incf counter, 1     ;increments the counter
114     movf counter, 0     ;updates the w value
115     subwf maxcounter, 0 ;maxcounter - counter
116     btfsc STATUS, 2     ;[maxcounter - counter] == 0 ?
117                         ;(2nd bit=zero bit : result of the ALU, p31 in datasheet)
118     clrf counter        ;counter <- 0
119
120 clear:
121     movlb 00h
122     bcf PIR1, 0         ;clear timer1 interrupt
123     bsf T1CON, 0        ;timer1 on
124     RETFIE
```

Code 3: Test of the 7 segments

# 5 Software

We decided to keep the software architecture that we presented in our previous report: the round-robin with interrupts. Meanwhile, as suggested in the feedback of the previous report,

we will not trigger our interrupts directly using an external source. We will instead periodically monitor the state of our infrared receiver.