INFO-2055: Embedded Systems Project

Customer Counter
Final report

Crucifix Arnaud	Goffart Maxime	Joris Olivier
170962	180521	182113

Academic year 2020 - 2021

Contents

1 Introduction **2**

2 Project idea **2**

3 Composition and functioning of the project **2**

4 Components **3**
- 4.1 Components for the receiving circuit 3
- 4.2 Components for both emitting circuits 3

5 Schematics **4**
- 5.1 Receiving circuit 4
- 5.2 Emitting circuit 5

6 Software architecture **6**

7 Code organisation **7**

8 Assembly code **10**

9 Demonstration **10**

10 Conclusion **10**

1 Introduction

In this report, we will present our project idea. Then, we will list the components that have been used in order to achieve the intended goal, including explanations on why we chose what we chose. Afterwards, we will explain, based on a schematic, how we connected all these components together. Subsequently, we will provide details on the software architecture we picked as long with the structure of the code. In the end, we will describe what we have learned throughout this project.

2 Project idea

Our idea is to build an automated customer counter for shops based on a infrared system. This idea was inspired by the actual sanitary situation that has been ongoing for more than a year. Currently, shops are restricting the number of trolleys or they are putting an employee at the entrance whom is regulating the flow of customers. In our opinion, these solutions are not very efficient. Thus, we want to use technology to improve the situation.

3 Composition and functioning of the project

Our system will be composed of 3 major sub-systems: 2 emitting circuits (Figure 3) and a receiving circuit (Figure 2). They will be placed the following way:

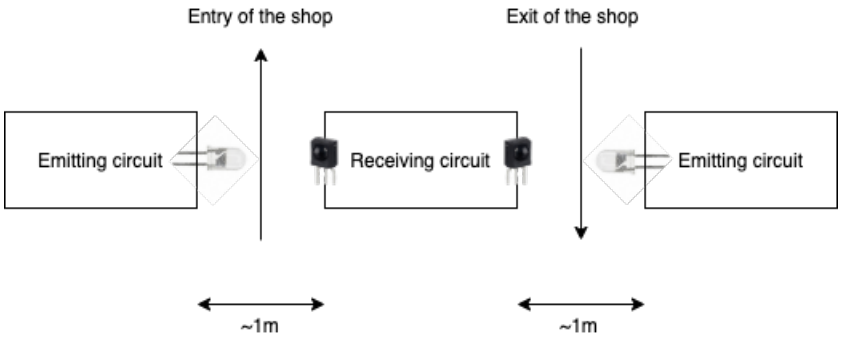

Figure 1: Positioning of the sub-systems

An emitting circuit is an infrared emitter that emits IR signals at a frequency of 38kHz. The central receiving circuit will be composed of a IR receiver on each side (left and right). Moreover, it will have buttons to set the number of customers that can enter and two 7-segments to display the number of people that can still enter the shop.

The first step is to choose the number of customers that can enter the shop with the buttons that are on the receiving circuit. After validation of this number, the system will be operating. If someone enters the shop, the counter is decremented[1]. If someone leaves the shop, the counter is incremented. In each case, the number displayed on the 7-segments will be updated accordingly to the new value of the counter. During the "operating mode", it is possible to go back to the configuration mode by using a dedicated button on the receiving circuit.

[1]Reminder: the counter represents the number of people that can still enter the shop.

The detection of someone entering or exiting will be done the following way. The emitters will continuously generate IR signals. When the receivers, on the receiving circuit, detects an IR signal, it outputs a voltage close to GND (OV). When it does not detect an IR signal, it outputs a value close to the voltage on the pin 3 which is the V_{dd} of the circuit (approximately 5V). Thus, if the voltage at the output of the receivers is close to 0V, we can conclude that no one has entered or left (depending on the receiver). Similarly, if the voltage is close to 5V, we can conclude that someone has entered or left.

4 Components

In this section, we will list all the components that we used for our project and explain the choices that have been made for the most critical ones. The references, in the following tables, are clickable and they refer to the shop, where we bought the components, which has the datasheet of each component that we used.

4.1 Components for the receiving circuit

The components for the receiver circuit (Figure 2) are listed in the table Table 1.

Description	Reference	Quantity	Usage	On the schematic
MCU	PIC16F1789	1	Controls the circuit	MCU1
IR receivers	TSOP58438	2	Detect IR signals	U4, U5
7 segments	LS0565SRWK	2	Indicate number of people that can still enter	U2, U3
Voltage regulator	L7805CV	1	Regulates tension from 9V to 5V	U1
Header	22-28-4060	1	Used to connect the PicKit3	H1
Buttons	Generic	4	Used to set the number of people that can enter the shop	B1→B4
Diode	1N4149	1	Prevents destroying the cirucit if the battery is put the wrong way	D1
Green LED	HLMP-C515	1	Indicates status of the receiving circuit	D2
Resistor 100 Ohm	Generic	2	Resistors for the IR emitters	R20, R21
Resistor 220 Ohm	MCRE000029	15	Resistors for the different LEDs (7 segments & green LED)	R5→R18, R22
Resistor 10 kOhm	MF50 10K	5	Resistors for buttons and connection to the PIC	R1→R4, R19
Capacitor 100nF	SR201C104KAR	2	Decupling capacitors for the MCU	C3, C4
Capacitor 10μF	ECEA1HN100U	1	Used with the tension regulator to have a clean signal	C2
Capacitor 100μF	ECEA1VN101U	1	Used with the tension regulator to have a clean signal	C1
9V battery holder	MP000352	1	Used to connect the battery to the circuit	/
9V battery	Generic	1	Power the circuit	/
Cable	R2651DTSY16AC85	x	Used to connect everything together	/

Table 1: Components for the receiver circuit

For the MCU, we decided to stick with the one used for the laboratories of the theoretical course because it matches our needs for this project.
Regarding the 7 segments, we picked the Multicomp LS0565SRWK because we wanted red 7 segments with a common cathode.
Concerning the IR receivers, our first idea was to use phototransistor. But they were very sensible to ambient light resulting in our circuit not working has expected. After a few emails exchanged with Mr Fonder, he recommended to use TSOP receiver with modulated IR signals. We chose the Vishay TSOP58438 because, according to their datasheet, they have improved immunity against noise and an included preamplifier.
The other components are more common.

4.2 Components for both emitting circuits

The components for both emitter circuits (Figure 3) are listed in the table Table 2.

Description	Reference	Quantity	Usage	On the schematic
IR emitters	TSHF5410	2	Produce IR signals	D4
Timer 555	TLC555IP	2	Generate 38kHz signals for emitters	U6
Diode	1N4149	2	Prevent destroying the cirucits if the batteries are put the wrong way	D3
Resistor 68 Ohm	Generic	2	Resistors for IR emitters	R25
Resistor 1k Ohm	MCF 0.5W 1K	2	Used with the timers to get a frequency of 38kHz	R23
Resistor 20k Ohm	MF50 20K	2	Used with the timers to get a frequency of 38kHz	R24
Capacitor 1nF	SR215C102KAR	2	Used with the timers to get a frequency of 38kHz	C7
Capacitor 0,01μF	SR215C103KAR	2	Used with the timers to get a frequency of 38kHz	C6
Capacitor 100μF	ECEA1VN101U	2	Used to have a clean input signal	C5
9V battery holders	MP000352	2	Used to connect the batteries to the circuits	/
9V batteries	Generic	2	Power the circuits	/
Small prototype boards	MC01010	2	To hold all the components of the circuits	/
Cable	R2651DTSY16AC85	x	Used to connect everything together	/

Table 2: Components for both emitting circuits

For the IR emitters, we picked the Vishay TSHF5410 because they have a wavelength of 890 nm which works well with the receiver we chose.
We needed the 555 timers in order to generate a signal with a frequency of 38kHz for the emitters because the receivers (TSOP58438) detects IR signal with a frequency of 38kHz.
The other components are more common.

For all the components, including the ones which are more common, we made our selection based on multiple factors including availability when ordering, minimum quantity, and price.

5 Schematics

As explained in the 3^{d} section, our system is composed of three main sub-systems: 2 emitting circuits and one receiving circuit.

5.1 Receiving circuit

The receiving circuit is represented on the Figure 2. It is composed of several main components that will be described in this subsection.

The 7-segments displays U2 and U3 respectively display the tens and units of the remaining number of people allowed to enter.

To allow the user of the system to know if the circuit is in the configuration mode or in the operating mode, the D2 LED can be observed.
If it is staying switched on, the system is in the configuration mode: the user of the system has to set up the maximum number of clients allowed to be in the shop at the same time using buttons B2 to B4.
In the other case, the system is in the operating mode: clients can pass the two gates and the number of allowed clients to enter will be updated accordingly.

In order to configure and reset the circuit, the buttons B1 to B4 can be used. Each have their own purpose:

- The **B1** button allows the user of the system to reset it. The system goes back to the configuration mode when its is operating. If the system is already in the configuration mode, this button does not have an effect.

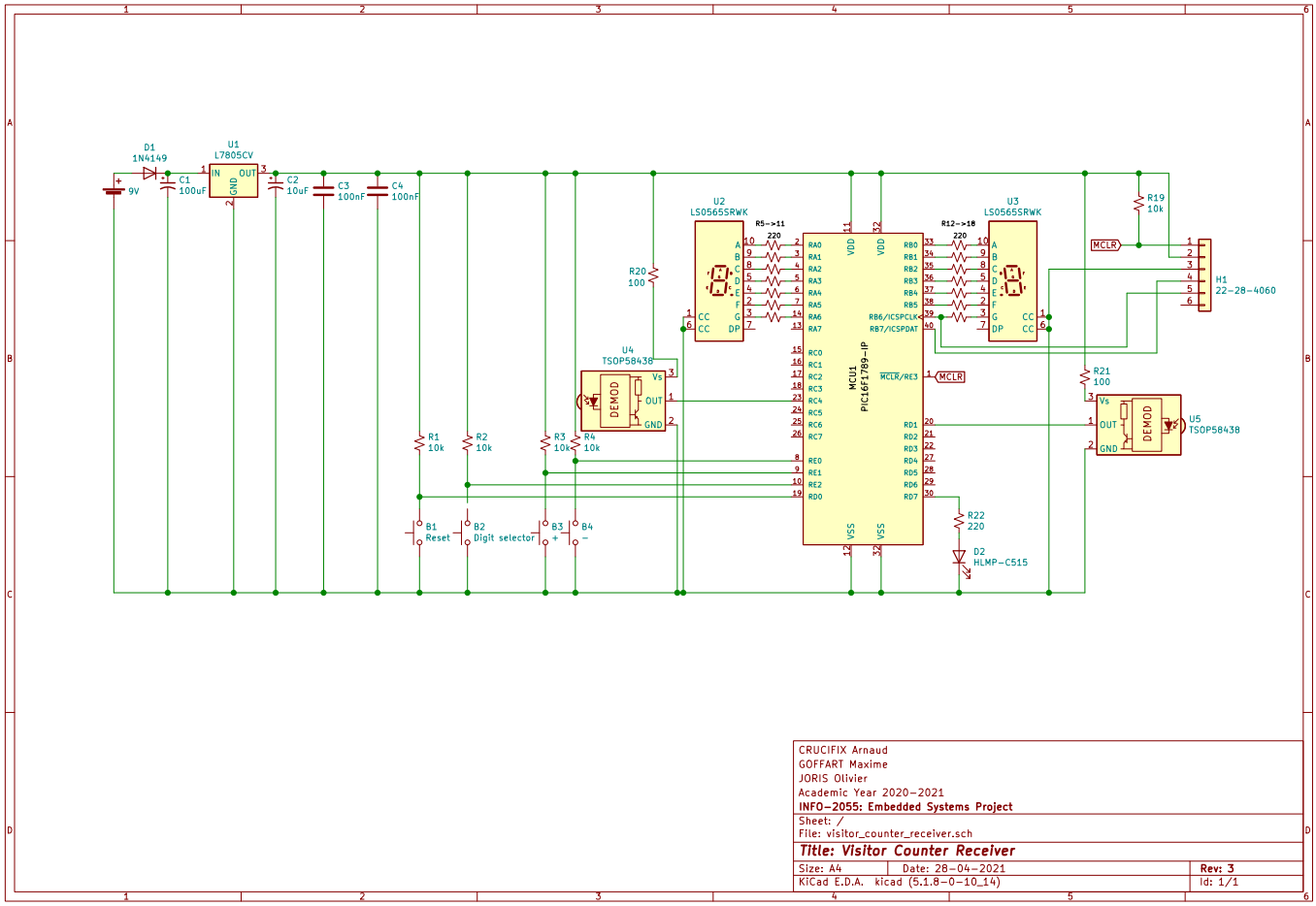

Figure 2: Schematic of the receiving circuit

- The **B2** button allows the user to choose the digit, represented on one of the two 7-segments displays, of the maximum number of clients to be modified by the B3 and B4 buttons.
- The **B3** button allows the user to increment the actual digit that has been selected using the B2 button.
- The **B4** button allows the user to decrement the actual digit that has been selected using the B2 button.

The two IR-receivers U4 and U5 are intended to respectively detect if someone entered or left the shop in order to update the 7-segments displays accordingly. We interface these two IR-receivers with the microcontroller using Schmitt triggers with CMOS levels.

The header H1 is used to connect the PicKit3 to the MCU in order to configure the latter. Explanations of the utility of the other components is available in the Table 1.

5.2 Emitting circuit

The emitting circuit is represented on the Figure 3.

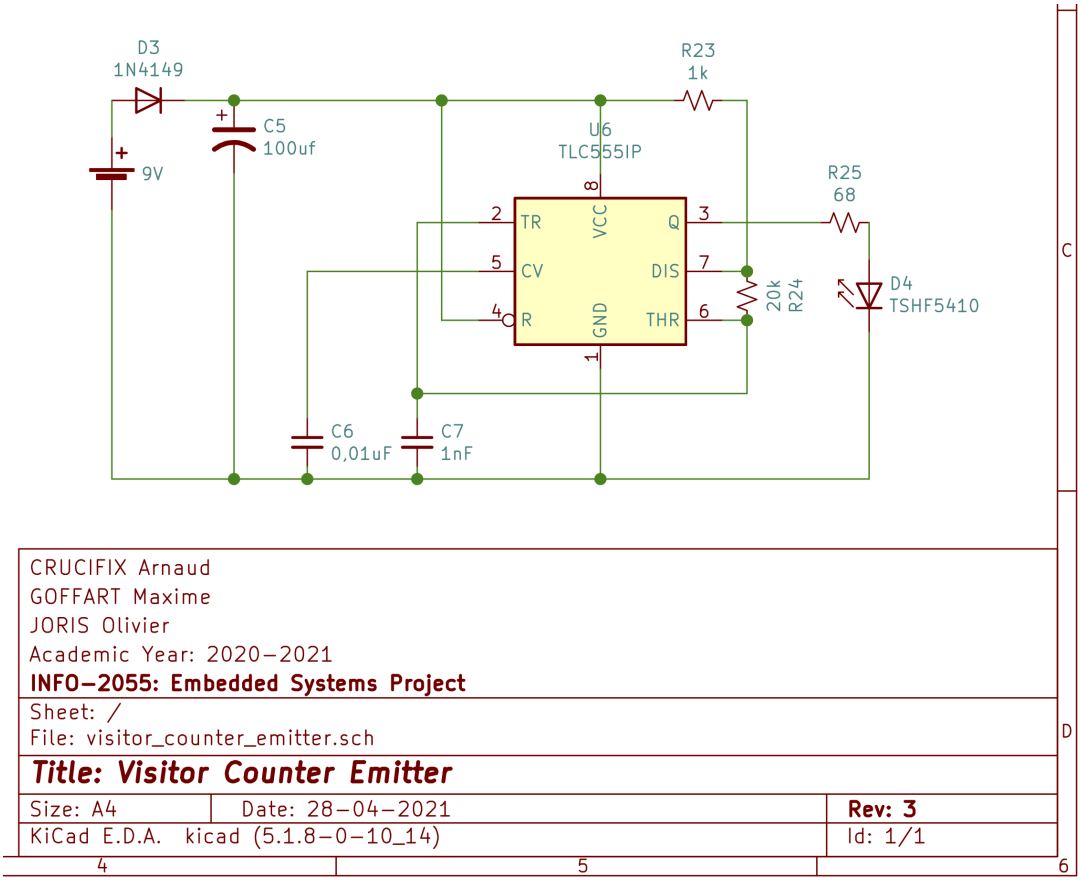

Figure 3: Schematic of the emitting circuit

We have two instances of this second circuit. One at the entry of the shop and one at the exit of it. They will continuously generate IR signals, as explained in the 3^{d} section.

Each emitter circuit contains an IR emitter D4 which is intended to generate IR signals. It also uses a timer 555 U6 used to generate a signal with a frequency of 38kHz for the IR emitter because the receiver on the other side detects IR signals at that particular frequency. Explanations of the utility of the other components is available in the Table 2.

6 Software architecture

In order to realise this project, we picked the round robin with interrupts architecture. Indeed, the round robin architecture was not suitable for us because we needed interrupts which are used to check the state of the 2 barriers at a high frequency. Consequently, the operations carried out in these interrupts are of short duration. In addition, because we wanted to use the simplest architecture suitable for our project, we came to the conclusion that the usage of a more complex architecture was not necessary. We do not need the notion of priorities between tasks.

The interrupts occur at a frequency of 2Hz.

When the receiving circuit is in the configuration mode, the following tasks are carried out:

1. Set up the maximum number of customers that the shop can accept based on the values set by the manager with the buttons B2 to B4. The detection of whether a button has been pressed or not is handled through an interrupt.

2. Update the values being displayed on both 7-segments only if a button has been pressed. This is not handled in an interrupt because we wanted to keep the interrupt as simple as possible.

3. Switch to the operating mode when both digits have been configured. As for the first task, the detection of whether a button has been pressed or not is handled through an interrupt.

When the receiving circuit is in the operating mode, the following tasks are carried out:

1. Check the state of the entry barrier and decrease by 1 the number of customers that can still enter if a crossing is detected.

2. Check the state of the exit barrier and increment the allowed number of customers to enter if a crossing is detected.

3. Check if the reset button has been pressed.

4. Update the values being displayed on both 7-segments only if someone has enter or leave the shop.

As explained previously, the first 2 tasks, in the previous enumeration, are carried inside interrupt routines. The last 2 tasks are not in interrupt routines because we wanted to keep them as simple as possible.

7 Code organisation

The organisation of our code can be described by the following skeleton in C-like pseudo-code:

```
/* Current phase of the system:
    - false if we enter in the setup mode
    (the manager chooses the maximum allowed customers than can enter);
    - true otherwise.
*/
volatile  bool  mode = 0;
/* Current digit selected:
    - false if the ten one is selected;
    - true if the unit one is selected.
*/
volatile  bool  selectedDisplay = 0;
// Displays the digits newly updated on 7-segments if true.
volatile  bool  needUpdateDisplay = 0;

// Value chosen by the manager during setup.
volatile  int  limitTenDigit = 0;
volatile  int  limitUnitDigit = 0;

// Current value.
volatile  int  tenDigit = 0;
volatile  int  unitDigit = 0;

// Maximum value that limitTenDigit, limitUnitDigit, tenDigit and unitDigit cannot exceed.
volatile int maxDigit = 9;
```

```
// Triggers periodically - only if mode == 0.
interrupt void setup(){
    switch(!! Button pushed){
        case(minus) :
            if(selectedDisplay == 0){
                if(limitTenDigit > 0){
                    --limitTenDigit;
                    needUpdateDisplay = 1;
                }
            }else{
                if(limitUnitDigit > 0){
                    --limitUnitDigit;
                    needUpdateDisplay = 1;
                }
            }
        break;

        case(plus) :
            if(selectedDisplay == 0){
                if(limitTenDigit != maxDigit){
                    ++limitTenDigit;
                    needUpdateDisplay = 1;
                }
            }else{
                if(limitUnitDigit != maxDigit){
                    ++limitUnitDigit;
                    needUpdateDisplay = 1;
                }
            }
        break;

        // Decides which one of the two 7-segments is going to be modified.
        case(next_digit) :
            if(selectedDisplay == 0)
                selectedDisplay = 1;
            else{
                tenDigit = limitTenDigit;
                unitDigit = limitUnitDigit;
                mode = 1;
            }
        break;
    }
}

// Triggers periodically - only if mode == 1.
interrupt void checkBarrier(){
    if(!! Pass detected at shop entry){
        if(unitDigit == 0){
            // 00
            if(tenDigit == 0)
                return;
            else{
                --tenDigit;
                unitDigit = maxDigit;
                needUpdateDisplay = 1;
            }
        }else{
            --unitDigit;
            needUpdateDisplay = 1;
```

```
        }
    }

    if(!! Pass detected at shop exit){
        if(tenDigit == limitTenDigit){
            // Maximum number of customers allowed by the manager reached.
            if(unitDigit == limitUnitDigit)
                return;
            else{
                ++unitDigit;
                needUpdateDisplay = 1;
            }
        }else{
            if(unitDigit == maxDigit) {
                if(tenDigit == maxDigit)
                    return;
                else{
                    ++tenDigit;
                    unitDigit = 0;
                    needUpdateDisplay = 1;
                }
            }else {
                ++unitDigit;
                needUpdateDisplay = 1;
            }
        }
    }
}

void update_display(){
    !! Updates the 7-segments based on tenDigit and unitDigit.
    needUpdateDisplay = 0;
}

void update_setup_display(){
    !! Updates the 7-segments during the setup phase based on
       limitTenDigit and limitUnitDigit.

    needUpdateDisplay = 0;
}

void main(){

    for(;;){
        if(mode == 0){
            !! Handles the buttons B1->B4 during the selection of the
               maximum number of customers.

            if(needUpdateDisplay == 1)
                update_setup_display();
        }else{
            if(!! Reset button pushed){
                mode = 0;
                tenDigit = 0;
                unitDigit = 0;
                selectedDisplay = 0;

                update_setup_display();
            }
            if(needUpdateDisplay == 1)
```

```
                update_display();
            }
        }
}
```

Code 1: Pseudo-code

8 Assembly code

The MPLABX project, including the assembly code based on the skeleton provided in the previous section, is available at the following link: MPLABX archive.

9 Demonstration

Due to the current sanitary situation, we do not know if we are going to be able to demonstrate our working project to you. Thus, we decided to make a small demonstration by video. This video is available at the following link: video.

10 Conclusion

In conclusion, after some troubles in choosing our components and making everything work as expected, the circuit is working as intended. We are able to detect people entering and leaving a shop while keeping track of how many people can still enter.

Finally, we want to mention a few thinks that we have learned.
Thanks to this project, we learned, with an example, that with a simple microcontroller, worth only a few euros, we can already build an interesting project with real-world usage.
We also learned that using infrared signals was not as easy as we thought. Our first idea was to use a simple infrared diode with a photodiode. Yet, during the presentation of our idea at the beginning of the first semester, you recommended to use phototransistors instead. However, it will later turn out that they were very sensible to ambient light. After a few emails exchanged with Mr Fonder, he suggested to use TSOPs with infrared signals modulated at 38kHz. So, we went that way and it turned out to be the best solution.