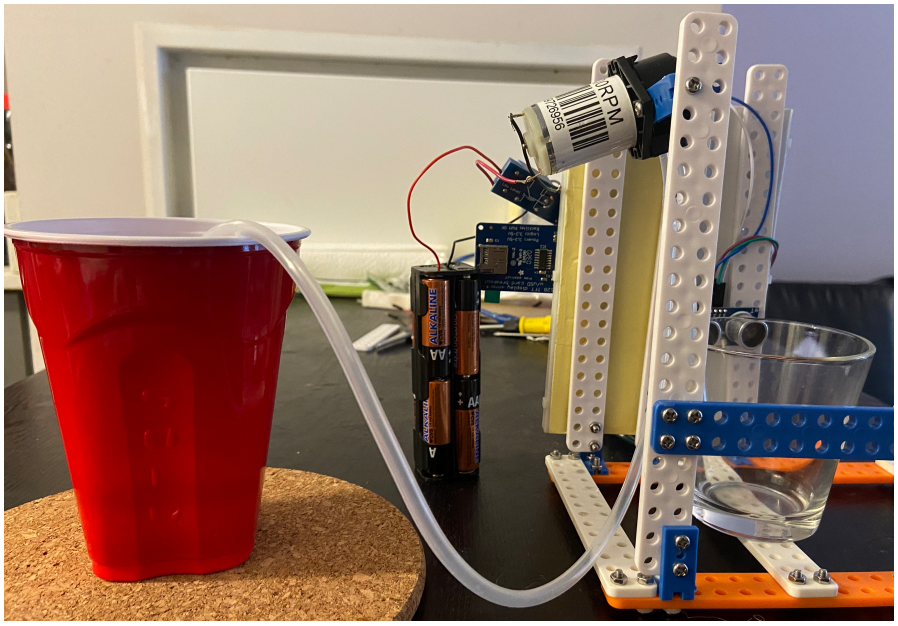
Multi-functional Water Dispenser

Shuyi Ying
Xiyue Luo

Demo link here : https://www.youtube.com/watch?v=0AsXo-FNhpo

Github link here : https://github.com/XiyueLuo/ESE519_FinalProject_Team9

Devpost Link here : https://devpost.com/software/multi-functional-water-dispenser



Final Project, Fall 2020
ESE519/IPD519: Real-Time and Embedded Systems
University of Pennsylvania

Table of Contents

Abstract

In this project, a multi-functional water dispenser is designed and successfully developed based on Arduino uno, together with general codes compatible to various types of MCUs. This water dispenser is an all-in-one portable device, with its basic water dispensing function, hourly reminding function, fluid intake recording and daily dehydrated warning functions, and real time and water volume information display function.

To achieve these fantastic functions, lots of electronic parts are applied and well organized. For sensors, an ultrasonic sensor detects the approaching cup. A real time circuit communicates and provides current time information to MCU. For actuators, a pump charged by 9-12V batteries are controlled by a relay, transferring the water in a container to the cup, a buzzer and a LCD.

To realize human computer interaction, a passive buzzer plays various lovely music segments. It reminds the user to drink every hour and warns at the customized time every day if the recorded fluid intake level is lower than the required drinking amount. The reminding interval, daily warning time and liquid drinking goal can be set by users. To visually present the information, there is a LCD screen showing the current time, fluid intake each time, accumulated water drinking amount, and the fully hydrated goal.

The original idea turns out to be a useful gadget, helping people live with a healthy lifestyle.

Motivation

To stay hydrated, having plenty of drinks is a daily necessity. However, as Harvard Health Publishing states, many of us aren't getting enough to drink, especially older adults.[1] "Older people don't sense thirst as much as they did when they were younger. And that could be a problem if they're on a medication that may cause fluid loss, such as a diuretic," says Dr. Julian Seifter, a kidney specialist and associate professor of medicine at Harvard Medical School.[1]

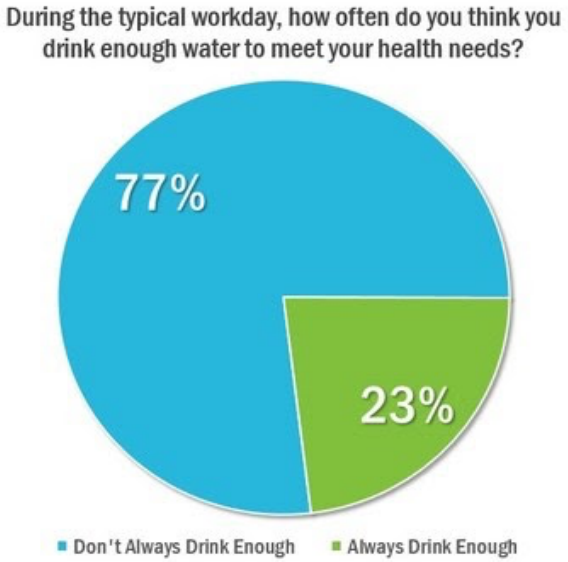


Fig 1 During the typical workday, how often do you think you drink enough water to meet your health needs?[2]

Not only for old adults, as figure 1 shows, the Quench study found that more than three-quarters (77%) of surveyed people do not think they consume enough water on a daily basis to meet their health needs[2]. Nevertheless, water acts as a significant role in our daily life. “Think of water as a nutrient your body needs that is present in liquids, plain water, and foods. All of these are essential daily to replace the large amounts of water lost each day," says Joan Koelemay, RD, dietitian for the Beverage Institute, an industry group.[3] Kaiser Permanente nephrologist Steven Guest, MD, agrees: "Fluid losses occur continuously, from skin evaporation, breathing, urine, and stool, and these losses must be replaced daily for good health[3]."

The issue that is revealed by the above data and research also perplexed our students in daily life. My teammate and I find it is hard to drink on a regular basis during extremely busy final weeks. It is often the case that we get dehydrated and are bothered putting brakes on the ongoing work to drink water, which leads to body unwell and several physical problems. To make a rehydrating habit, a lot of methods have been tried, i.e.,

using an App to record fluid intake and setting alarms for reminds, but it turns out not to be an efficient solution.

Lack of corresponding effective solutions, we are motivated by the ambition to address such an issue for a wide range of people, taking care of their body health. We decide to design a multi-functional water dispenser, a small, portable device that can be placed on the desk. Once this gadget reminds to have liquid, users can take action immediately, without wasting time to fetch water and being distracted. Additionally, it is able to record user drinking amount and let them know when they still need liquid. We believe that this multi-functional water dispenser can definitely help people to live in a healthier lifestyle.

Goals

The goal of this project is to develop a multi-functional water dispenser, with the basic function of dispensing water automatically when a cup approaches. Advanced functions include reminding users to drink water hourly, recording daily fluid intake and warning when it is below the required level, displaying real time and water amount information, which are realized by a system consisting of an Arduino uno, a buzzer, a LCD screen, a relay, a pump, batteries, and an ultrasonic sensor, real time circuit (RTC), together with a model to place and connect all the parts.

A. Milestone 1

For the first 3 weeks, the goals are realizing the following functions of water dispenser:

1. Dispense water when a cup approaches.
2. Write the library of real time circuit DS1307.
3. Drive buzzer to play a melody.

B. Final Demo

For the following 2 weeks, the goals are realizing the following functions of water dispenser:

1. Record both the fluid intake each time and the accumulated daily water intake amount.

2. Allow users to set daily drinking level and a specific time when buzzer warns if the accumulated water drinking amount is under the required level.

3. Drive buzzer to warn at the set time and remind to drink water hourly with different music segments.

4. Display real time information, the amount of water dispensing each time and the amount of water that the user has already drunk during the day.

Methodology

A. Basic water dispensing function

To realize the basic water dispensing function, the infrared sensor is originally used to detect the approaching cup. The sensor outputs a voltage with high or low level to Arduino, indicating whether there is an obstacle breaking the beam or not. However, it turns out to be too sensitive, easy be disturbed by the environment disruptive factors, i.e., the infrared beams contained in the environment light. Additionally, it is not able to calculate the distance between the cup and the dispenser, so the threshold distance cannot be set to deal with the over-sensitivity issue.

The ultrasonic sensor replaces the infrared sensor because it can calculate the distance given the time information of trigger and echo waves. Based on that, we drive relay to let pump off when the distance is over 5cm and let pump open when the distance is below 5cm.To be more specific, when the measured distance is smaller than the threshold, i.e., 5 cm, the Arduino will control the actuator to dispense water.

For the actuator part, the water pump needs to work with the voltage domain between 9V and 12V. Therefore, batteries are applied to charge the pump. In order to control the pump, between the battery and the pump, a relay is added to control the connection and the disconnection of the circuit. The connection of battery, relay and pump is show in figure 2.

The left 3 pins of relay are controlled by the right 3 pins. The relay has the logical level of 5V, therefore, it should be connected to the 5V on Arduino and also GND on Arduino. PD4 is set as a digital output pin on Arduino, and when PD4 is high, the COM pin is connected to NO pin. Similarly, when PD4 is low, the COM pin is connected to NC pin. In this project, the pump only need to work when the distance is smaller than the threshold, so the pump is connected to the normally-closed pin, which is “NC”.

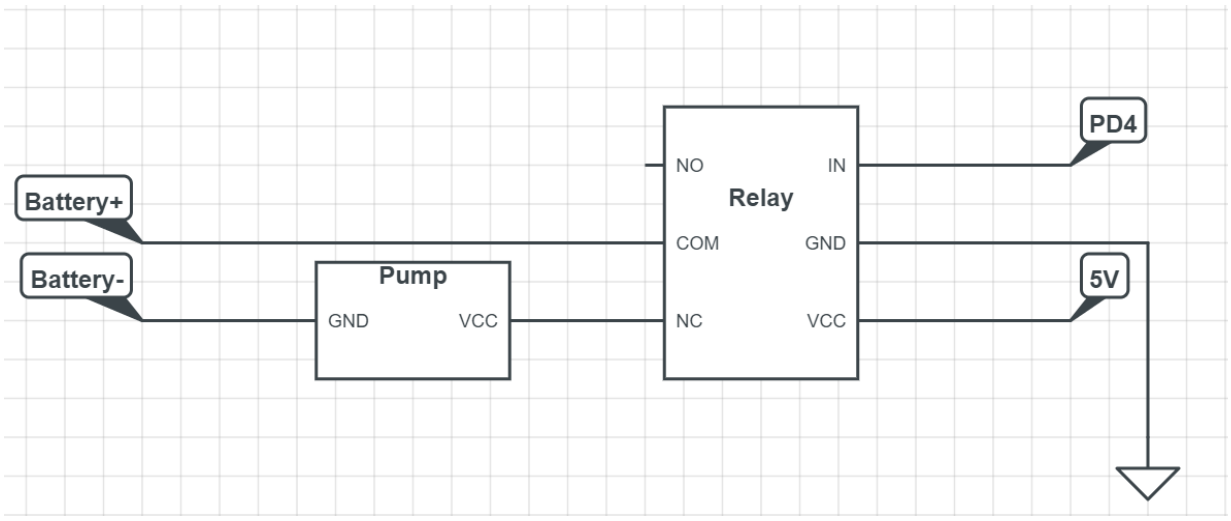


Fig 2 The connection of battery, relay and pump

When the distance is smaller than the threshold, Arduino sends signals to the relay to build the connection between battery and pump, and the pump begins to transfer water from the water container to the cup. When the distance is larger than the threshold, the circuit is disconnected, and the pump stops working.

B. Real time circuit (RTC) communication

To realize hourly reminding and daily warning functions, real time circuit is necessary since Arduino can only measure the relative time rather than the real time. Here, the communication between RTC and Arduino uses I2C protocol. I2C is an easy synchronized communication protocol since it only needs 4 connection wires, i.e., SDA, SCL, VCC and GND, for Arduino to receive the data from RTC. The connection of DS1307 is shown in figure 3.

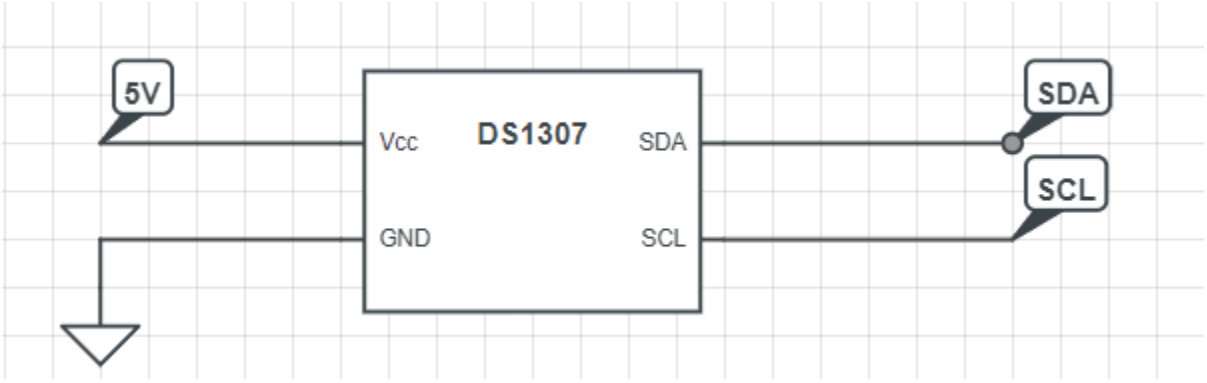


Fig 3 The connection of DS1307

In this project, DS1307 is used to get precise real-world time. Here, the I2C library is an external library, only the library of DS1307 is written by ourselves. DS1307 stores all the

time information inside its internal registers. And when we need this information, we can get it by reading data from DS1307 register. Therefore, the ds1307_read_register() function is used to get information from DS1307. In the function, the input parameter is the address of the register that we expect to get information from, and the output parameter is the received data. Inside the function body, firstly pass the input address of register to the device_data[0]. And then send read command to receive the data from DS1307 by using function i2cMasterSendNI() and i2cMasterReceiveNI() functions. The received data is also stored in device_data[0]. Finally, return the device_data[0] to get the final received data.

According to the datasheet of DS1307 shown in figure 4:

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

Fig 4 Datasheet of DS1307

For second information, it is stored in the register with address of 0x00. The MSB is used to represent CH, which is the clock halt (CH) bit. When this bit is set to 1, the oscillator is disabled. When cleared to 0, the oscillator is enabled. We do not need the CH bits when getting the time information, so the second should calculate by the formula:

$$second = (second \gg 4) \times 10 + (second \,\&\, 0b00001111)$$

Where second is the number stored in register with address 0x00 without the Bit 7.

Similarly, for the minute information, it can be calculated by the formula:

$$minute = (minute \gg 4) \times 10 + (minute \,\&\, 0b00001111)$$

Where minute is the raw number stored in register with address 0x01.

For hour information, it is stored in the register with address of 0x02. The format of hour should be judged firstly. If it is stored in 12-hour mode, Bit 6 will be high, and if it is stored in 24-hour mode, Bit 6 will be low.

In the 12-hour mode, Bit 5 represent the AM/PM, and high is PM. The hour can be calculated by the formula:

$$hour = (hour \gg 4) \times 10 + (hour \ \& \ 0b00001111)$$

Where hour is the number stored in register with address 0x02 without the Bit 7, 6 and 5.

In 24-hour mode, the hour can be calculated by the formula:

$$hour = (hour \gg 4) \times 10 + (hour \ \& \ 0b00001111)$$

Where hour is the number stored in register with address 0x02 without the Bit 7 and 6.

C. Different melodies for two situations

To distinguish the hourly reminding and daily warning, the buzzer should play different music segments with distinguished melodies.

When the buzzer does not need to ring, Arduino creates a PWM wave with frequency lower than 20Hz, which is over the audible frequency range of human beings. In this project, both OCRA and OCRB is set to 0 to mute the buzzer. To drive the buzzer work according to different situations, three self-defined sub functions are used, together with 4 arrays, 2 for the frequency and corresponding beats of reminding music and 2 for warning music. There is a delay function to simulate beats of a music segment. This means that the buzzer will keep ringing at different frequency with corresponding varied time period. Delay function achieves this aim with two idle ‘for’ loops, where the execution time of 1 loop is changed with the elements in the beat array and the time of the other loop stays constant to ensure the minimum delay time period. The local variables in delay function are both volatile long type, which allow them not to be optimized and let them to be regarded as constants.

The other two functions are reminding and warning functions. Each contains a ‘for’ loop to drive buzzer to play the corresponding melodies. They are included in different ‘if’ block to be triggered when it is appropriate, Afterwards, the buzzer will stop automatically when the music is over.

D. Fluid intake recording function

To measure the amount of water, the Arduino internal timer is used. Since the pump has a constant outgoing speed, which is 0.0012 milter per millisecond. The amount of water is calculated based on the pump working time. When the relay builds the connection, Arduino records TCNT and times of overflows until the relay disconnects the circuit.

Afterwards, the time period is calculated based on the recorded counter number. And the amount of water is derived by multiplying the time period and the water outgoing speed.

E. Time and water amount information display

In terms of LCD, it helps users read information such as time, the amount of water dispensing this time and the amount of water that the user has already taken during the day. In this part, the connection between LCD and Arduino is SPI bus, and our task is to write functions that can be used to display the information above.

Here, D/C pin of LCD should be connected to PB0 on Arduino by default. However, PB0 is the input capture interrupt pin of Timer 1, which is used to capture the echo signal of ultrasonic sensor. Therefore, D/C pin of LCD can move to other pin, i.e. PD2, and set that pin as a digital output pin. When PD2 is low, the LCD can treat the sending data as commands, and when PD2 is high, the LCD can treat the sending data as display information.

According to other parts of methodology, all the needed information has already obtained. The real time information can be obtained from a real time circuit, the amount of water dispensing this time can be calculated by Arduino, and the amount of water that the user has already drunk is the sum of fluid intake each time. After obtaining all needed information, the information can be transferred to the String format by using sprintf() function, and then using LCD_drawString() to display on the LCD.

Results

The goals are all successfully achieved and the final solution is presented in figure 5.

For sensors, we changed the infrared sensor to the ultrasonic senser to detect an approaching cup. A real time circuit DS1307 is used to communicate with the MCU and provide real time information. For actuators, a pump charged with 9-12V batteries is controlled by a relay. It can pump water from the container to the cup according to the commands of Arduino uno. A passive buzzer plays various music segments, serving as an hourly reminder and a dehydrated warner. To decide the conditions of driving buzzer to ring with different music, both the reminding time interval and the calculated fluid intake level are offered by the timers of Arduino uno. Additionally, a LCD ST7735R is to show the user-computer interaction information, i.e., current time, water volume taken this time, sum water volume this day, the required fluid intake level this day, and the daily drinking goal together with daily dehydrated warning time are set by users.

All the parts are well fastened and connected with a self-made shelf, allowing the whole design to be more portable and convenient for use.

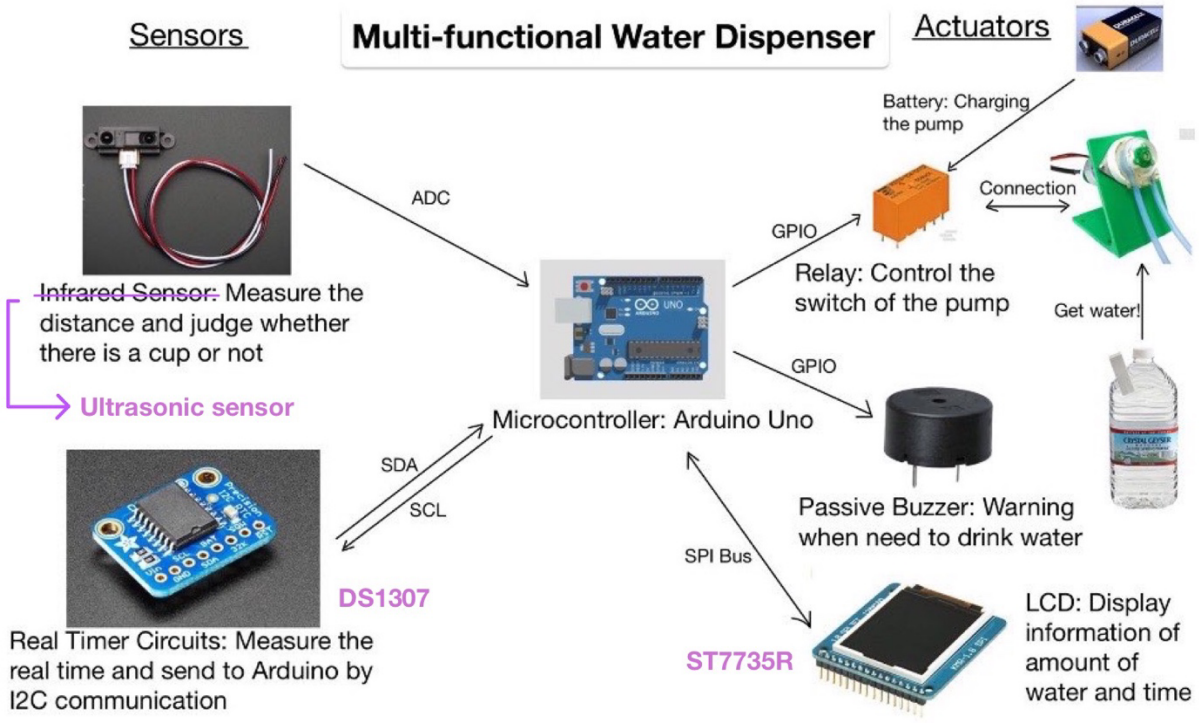


Fig 5 Final solution in this project

Conclusion

In this project, we learned how to choose and use a wide range of electronic parts properly, and how to connect and control them by Arduino uno. The interesting self-designed functions are well realized, original and practical.

The codes are written neatly and generally, not Arduino like codes, so that can be used in various MCUs. For example, a self-defined delay function is helpful to drive the buzzer to play different melodies. Idle ‘for’ loops and volatile local variables are a smart way to drive the buzzer to ring with a set of varying beats. This method deals with the issue that the _delay_ms() function can only receive constant parameters. We also encapsulated each function into their sub-function blocks and let main function to call them smoothly and effectively. In addition, RTC is most difficult part of this project, but we successfully accomplish the library and build the communication between RTC and MCU.

There are also some unexpected turns but we finally come up with our effective solutions. One of the pins of LCD ,i.e., PB0, is collided with the interrupt capture pin of the ultrasonic sensor. Since PB0 serves as a GPIO pin for LCD, we revised the LCD graphics library and use another pin PD2 to achieve the same aim. Additionally, we found that the water transferring speed is too fast and the volume values may be not precise. To solve this problem, we reset the resolution of the dispensing time to be finer.

In the future, advanced functions should be added. For instance, either a real keyboard or a virtual keyboard together with wireless communication between MCU and a smartphone app can be an interesting human-computer interaction design. Users can simply set the reminding interval, warning time, and desired drinking goal rather than revising the code. In addition, an exquisite model should be made, for a more fantastic look and higher quality. It can also protect the circuit not to be ruined by the splashing water.

References

1. https://www.health.harvard.edu/staying-healthy/how-much-water-should-you-drink

2. https://www.prnewswire.com/news-releases/nearly-80-percent-of-working-americans-say-they-dont-drink-enough-water-quench-survey-300668537.html

3. https://www.webmd.com/diet/features/6-reasons-to-drink-water#1

4. https://datasheets.maximintegrated.com/en/ds/DS1307.pdf

Appendix A

Appendix A1: Library of I2C

I2C is a two-wire bidirectional network, which is used to build transmission of information between a lot of devices and micro-controller units. I2C is a primary communication protocol, since a lot of common devices, i.e., temperature sensors, real time circuit, are using I2C protocol to communicate.

I2C uses only two wires (SDA and SCL) to build duplex communication between devices. I2C allow micro-controller units to having several devices on a single bus. Because I2C uses a 7-bit number address to identify device. However, it does not allow the micro-controller units to have more than $2^7 - 1 = 127$ devices on a single bus.

What's more, I2C requires two 4.7K pull-up resistors. For small numbers of devices (maybe less than 4 devices), it is enough to activate the internal pull-up resistors. To achieve this, set port of I2C pins (SDA/SCL) high.

The I2C library include s 2 files: "i2c.h" and "i2c.c".

In i2c.h, firstly call "i2cconf.h", which define the data buffer size is 0x20. This means that the maximum sending and receiving data size is 0x20. Then, it declare several important functions, such as i2cInitial(), i2cSendByte(), i2cReceiveByte(), i2cMasterSendNI() and i2cMasterReceiveNI(), and these functions are realized in i2c.c file.

For i2cInitial(), the purpose of this function is to initialize the i2c bus and start the data transmission. This function has both void input and return parameters. In the function body, firstly set up the internal pull-up resistor by setting PORT register high. Then, clear both slave transmit and receive handler, and set a unified bit rate for both master and slave at 100kHz. Finally, enable the two-wire interface, interrupt and slave acknowledgement and set the I2C bus idle (when data is transmitting, the status will be busy).

For i2cSendByte() and i2cReceiveByte(), is used to send a single byte to devices or receive a single byte from devices. In i2cSendByte(), the input parameter is the data that need to be sent. Also, in i2cReceiveByte(), the input is ACK flag, only if the ACK is true, the micro-controller unit will read data. If the ACK is false, the micro-controller unit will drop the data. In these two functions, the functions in AVR library is used, such as outb(), inb() and BV(). Here, outb() is used to write data to the IO port and inb() is used to read data from IO port. BV(x) is a function that is used to calculate, which equals to (1<<x).

For i2cMasterSendNI() and i2cMasterReceiveNI(), the purpose of these two function is to send I2C data to a device or receive I2C data from a device on the bus based on non-interrupt condition. In these two functions, firstly disable the interrupt and then start the

sending process to send the device's address and write or read command by using i2cSendByte() function. After that, check whether the device is online. If the device is online, the micro-controller unit will begin to send or receive data and ACK it one byte by one byte by using i2cSendByte() or i2cReceiveByte(). When the sending or receiving complete, stop the I2C communication and enable the interrupt.

Appendix A2: Library of ST7735

This library is used to drive the LCD. This library contains two files: "ST7735.h" and "ST7735.c".

In "ST7735.h", all the global variables are declared, including the pin number, error codes and register address. The default pin number is shown in figure 6,

pin on LCD	pin on Arduino Uno
LCD_RST	PORTB1
LCD_TFT_CS	PORTB2
LCD_MOSI	PORTB3
LCD_SCK	PORTB5
LCD_DC	PORTB0
LCD_LITE	PORTD6

Fig 6 Default pin information of LCD and Arduino Uno

However, in this project, PB0 is the input capture interrupt pin which is used to capture the echo from ultrasonic sensor. Therefore, the LCD_DC pin have to connect to another pin on Arduino. Firstly, we tried to connect the LCD_DC pin to PB4, but PB4 is the pin that can be used as MISO pin. Because the LCD has already defined the MOSI at PB3, the status of PB4 cannot change at will. Finally, we choose PD2, which pin does not have any other functions. The pin number we used in this project is shown in figure 7,

pin on LCD	pin on Arduino Uno
LCD_RST	PORTB1
LCD_TFT_CS	PORTB2
LCD_MOSI	PORTB3
LCD_SCK	PORTB5
LCD_DC	**PORTD2**
LCD_LITE	PORTD6

Fig 7 Revised pin information of LCD and Arduino Uno

In the "ST7735.h", some macro functions are also defined to bring convenience. In this library, set() equals to the bit operation $| = (1 \ll bit)$; clear() equals to the bit operation $\& = \sim(1 \ll bit)$; toggle() equals to the bit operation $\wedge = (1 \ll bit)$. loop_until_bit_is_set() and loop_until_bit_is_clear() will execute a void loop until the bit is set or cleared respectively.

There are several important functions in this library, including Delay_ms(), sendCommands(), lcd_init(), LCD_setAddr(), SPI_ControllerTx_16bit().

Delay_ms() is the function that has the same usage as _delay_ms() in “util/delay.h”. However, _delay_ms() only allows the input parameters of constant. If we want to use the delay function and the input is variable, a new Delay_ms() is built. Inside the function body, a while loop containing _delay_ms(1) will be executed n times, to delay n ms.

sendCommands() is the function that used to send commands to LCD. The input parameters of the function are a pointer of command’s address and an int of its length. The return of the function is void. Inside the function, sending the commands one byte by one byte. Also, clear the D/C pin when sending commands, and set the D/C pin when sending data. LCD used the status of D/C pin to judge the received bytes are commands or data.

lcd_init() is the function that used to initialize the LCD screen. In this function, firstly, set all pins as following: set all used pins as output pins and enable the internal the needed pull up resisters, use fast PWM mode in Timer 0 to generate a PWM wave at PD6 to set the initial brightness of LCD. Also, wait for 50ms and set the RST pin high to start the LCD. Then, initialize SPI controller register. After that, declare the list of commands that need to be sent. Finally, send all commands to LCD and finish the initialization.

LCD_setAddr(), is the function that used to set the memory address of each pixel on screen. The input parameters of the function are 4 integers which can represent the pixels on screen, and the output parameter is void. Inside the function body, send 3 commands to LCD: the column information, the page information and save these information to RAM.

SPI_ControllerTx_16bit() is the function that can send 16-bits data to LCD, which is always used as color information. The input parameter of the function is the data that need to be transmitted, and the output parameter is void. Inside the function body, firstly set the CS pin low to enable the serial interface and start data transmission. That save the first 8 bits in temp, cause SPI only allows transmit 8 bits in one time. Then transmit first 8 bits data and then the following 8 bits data to the register. When all the transmission is completed, set the CS pin high to disable the serial interface and stop transmission.