Project: Gyro RC car & IOT

Date: 2023-12-18

Author/Partner: Soon Ho, Lim / Tae Geon, Han

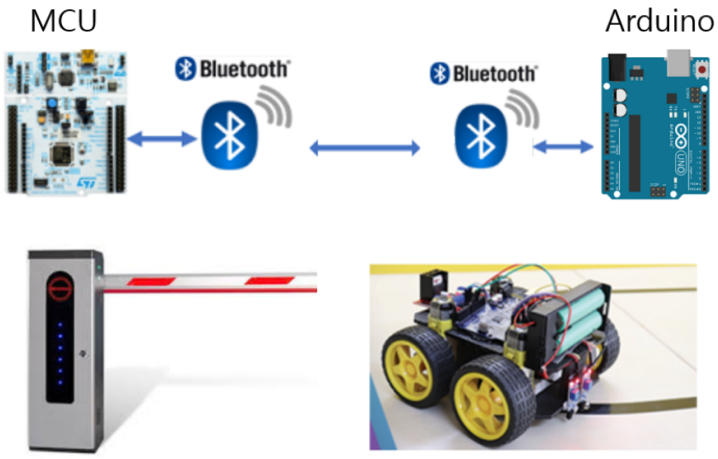
Github: https://github.com/hhangun/EC-taegeon-793

Demo Video: Link

I. Introduction

This project applied gyro sensor to the RC car controller in purpose of manipulate the RC car with difference of angles in x, y, z axis. Along with the RC car, smart house with gate, parking lot, house is made with 3 actuators and 4 sensors. There are 3 modes for the smart house: normal mode, security mode and power-saving mode.

Total of 4 MCU are utilized for this project: for the controller arduino uno R3, for the gate, the house and parking lot and the RC car, 3 STM32F411 are used.



Hardware

Item	Model/Description	Qty
MCU	NUCLEO -F411RE	3
Arduino	UNO -R3	1
Analog Sensor	CDS(SZH-SSBH-011)	1
Digital Sensor	Ultrasonic distance sensor(HC-SR04)	2
	Gyro sensor(mpu6050)	1
	PIR motion detector sensor(HC-SR501)	1
Actuator	DC Motor	2
	RC Servo Motor (SG90)	4
	Buzzer	1
Display	LED	3
Communication	Bluetooth Module(HC-06) / communication between the modules	2
Others	Actuator Controller : DC motor driver(L9110s)	1
-	-	-
Total	-	22

Software

- Keil uVision IDE
- CMSIS
- EC_HAL
- Arduino IDE

II. Problem

Problem Description

1. System Mode:

MODE	Description
Normal mode	It was updated for the first time and has not pressed the button, or if you press the reset button, it is in normal mode. Alternatively, pressing the button (A13 of MCU_1) switches to Normal mode in Saving mode.
Security mode	Pressing button (A13 of MCU_1) changes to Security mode from Normal mode.
Saving mode	Pressing button (A13 of MCU_1) converts to Saving mode from Security mode.

2. Parking lot

MODE	Description
Normal Mode	When a vehicle enters the parking lot: If it comes within a specified distance (20cm), a buzzer emits a sound. When the driver exits the vehicle to walk towards the house: motion detected by a sensor between the house and the parking lot triggers the LED to illuminate for a set duration and automatically opens the house door. Subsequently, the LED turns off, and the door closes automatically. If the CDS sensor detects bright light: the parking lot roof closes. If the CDS sensor detects darkness: the parking lot roof opens.
Seccurity mode	It's a mode for when someone breaks into a house. When the motion sensor between the parking lot and the house detects the motion: The roof of the parking lot and a house door are closed, the LED is blinking, and the buzzer sounds. At this time, the roof is closed regardless of the CDS sensor.
Saving mode	It is a mode for saving electricity. When a vehicle enters the parking lot: If it comes within a specified distance (20cm), a buzzer doesn't make a sound unlike a normal mode. When the driver exits the vehicle to walk towards the house: When a movement is detected, the house door opens, but the LED does not turn on. In the saving mode, the ON/OFF of the roof using the CDS sensor is not operated to reduce unnecessary consumption and is always closed. In addition, like in normal mode, the open door is closed after a certain period of time.

3. Entrance:

MODE	Description
Normal Mode	When no vehicle is detected: the gate remains closed, and a red LED is illuminated. When a vehicle is detected within a given distance from the entrance gate: the gate opens, and a green LED is activated. After the gate opens and the vehicle passes through: the gate closes, and the LED transitions from green to red.

4. RC car

MODE	Description
Normal mode	The values are transmitted between the MCU (TX:PA10 RX:PA9) and ARDUINO (TX:8 RX:9) using Bluetooth. It transmits the value of the gyro sensor and with this value, it is possible to move in a desired direction by adjusting the servo motor of the RC Car.

5. Sensor Unit:

(1) MCU_1 (Parking lot)

Function	Sensor	Configuration	Comments
Car detect	Ultrasonic distance sensor	Sampling 0.05 sec Check object presence within 20cm generates	Need to check for outlier measurements (at least 7/10 Postive) VISITOR_LOG under SECUR_MODE
Warning of close distance from the wall	Buzz	Buzz state generates delay 0.5 seconds 5 times	
Light detect	CDS(Photoregister)	Trigger_TIM3, 1msec, RISE edge	The presence or absence of the roof is divided by the sensor value 1300.
Rotate roof and door	Servo motor	PWM Period: 20 msec	Each servo motor might be different, but they all share the same conditions regarding PWM.
Moving detect	PIR motion detector sensor	Push-Pull, Pull-up-Pull-down	
Light Up	LED	Push-Pull, Pull-up-Pull-down	ON/OFF RED

(2) MCU_2 (Entrance)

Function	Sensor	Configuration	Comments
Car detect	Ultrasonic distance sensor	Sampling 0.05 sec Check object presence within 20cm generates	Need to check for outlier measurements (at least 7/10 Postive) VISITOR_LOG under SECUR_MODE
Rotate door	Servo motor	PWM Period: 20 msec	Each servo motor might be different, but they all share the same conditions regarding PWM.
Light Up	LED	Push-Pull, Pull-up-Pull-down	ON/OFF RED, BLUE

(3) MCU_3 (RC Car)

Function	Sensor	Configuration	Comments
Communication with Arduino	Bluetooth(HC-06)	Baud Rate: 9600	Two bluetooth module must be communicating. This can be done by selecting switching the module as master ans slave by AT commands.
Rotate the wheels	DC Motor	PWM Period: 200 usec	from the gyro sensor, the signals will be received and comparing the signal with arranged values the DC motor should turn. 4 state there is: forward, forward faster, backward backward faster and last stop.
Control direction of RC car	Servo Motor	PWM Period: 10 msec	similar with the DC motor, the comparing the gyro sensor(Z-axis) with stated value the servo motor turns and gives angles for the RC car to turn left or right.

(4) Arduino

Function	Sensor	Configuration	Comments
Communication between MCU	Bluetooth(HC-06)	Baud Rate: 9600	Two bluetooth module must be communicating. This can be done by selecting switching the module as master ans slave by AT commands.
Control direction of RC Car by using Gyro sensor	Gyro sensor(MPU6050)	Interrupt Pin Number: 2	The value changes over time, so it should be used in consideration of this point when using a gyro sensor.

MCU Configuration

(1) Parking Lot and Smart House

Function	Register	PORT_PIN	Configuration
System Clock	RCC	-	PLL 84MHz
delay_ms	SysTick	-	500 msec delay is given for blinking LED and buzzer.
USART for check value through PC	USART2	TXD:PA_2,RXD:PA_3	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate
Ultrasonic sensor (timer interrupt)	TIMER4	ECO: PB_6	ICAP counter step time as 10 usec TIM4_CH1 as IC1 , rising edge detect TIM4_CH2 as IC2 , falling edge detect
Ultrasonic sensor (trigger pulse)	PWM	TRIG:PC_7	PWM of 50ms period. PWM pulse width of 10us
CDS sensor (Analog digital convert)	ADC	PB_0	Priority 2 TIM3, 1msec, RISE edge
servo motor/ roof	PWM	PA_1	20 msec PWM period
servo motor/ door	PWM	PA_0	20 msec PWM period
buzzer	GPIO	PA_8	OUTPUT, Pull Down, Push Pull
LED	GPIO	PA_9	OUTPUT, Pull Down, Push Pull
button	GPIO	PC_13	INPUT, Pull Down
External interrupt	EXTI	PC_13	GPIOC Pin: 13 trigger: falling priority: 0
Motion detect sensor	GPIO	PA_6	INPUT, Pull Down

(2) Entrance

Functions	Register	PORT_PIN	Configuration
System Clock	RCC	-	PLL 84MHz
delay_ms	SysTick	-	waiting for the car to pass the gate 1sec 3 sec delay before starting the program
Ultrasonic sensor (timer interrupt)	TIMER4	ECO: PB_6	ICAP counter step time as 10 usec TIM4_CH1 as IC1 , rising edge detect TIM4_CH2 as IC2 , falling edge detect
Ultrasonic sensor (trigger pulse)	PWM	TRIG: PC_7	PWM of 50ms period. PWM pulse width of 10us
USART for check value through PC	USART2	TXD:PA_2,RXD:PA_3	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate
servo motor for the gate opening	PWM	PA_1	20 msec PWM period

(3) RC car

Functions	Register	PORT_PIN	Configuration
System Clock	RCC	-	PLL 84MHz
DC motor A and B	PWM	motorA: PA_0 motorB: PA1	PWM period 200 usec
servo motor	PWM	PB_6	PWM period 10 msec
direction pin	GPIO	motorA: PC_10 motorB: PC_12	OUTPUT, No Pull Up Pull Down, Push Pull
USART for check value through PC	USART2	TXD:PA_2,RXD:PA_3	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate
Bluetooth	USART1	TXD: PA9 RXD: PA10	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate AT command mode: Slave

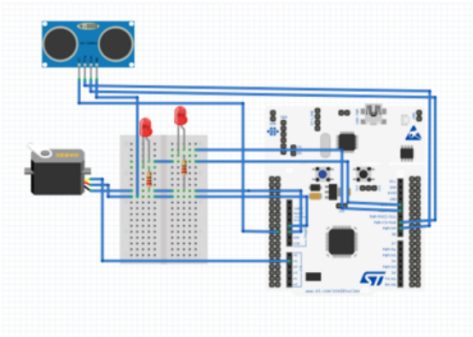
(4) RC car controller (Arduino uno R3)

Functions	Register	PORT_PIN	Configuration
Bluetooth	USART1	8 and 9	8-bit Data, 1-bit Stop bit 9600 baud-rate
Gyro sensor	I2C	SCL: A5 SDA: A4	9600 baud-rate

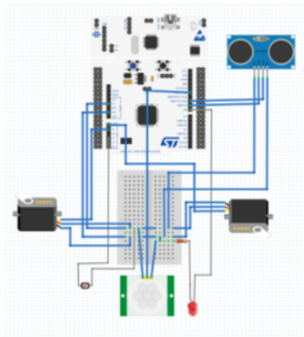
Circuit Diagram

This are the four MCU's circuit diagram

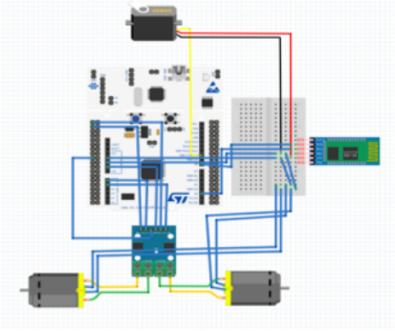
MCU (Entrance)



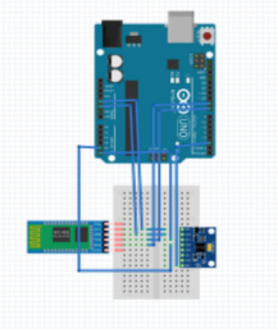
MCU (Parking lot)



MCU (RC car)



Arduino (Gyro sensor)

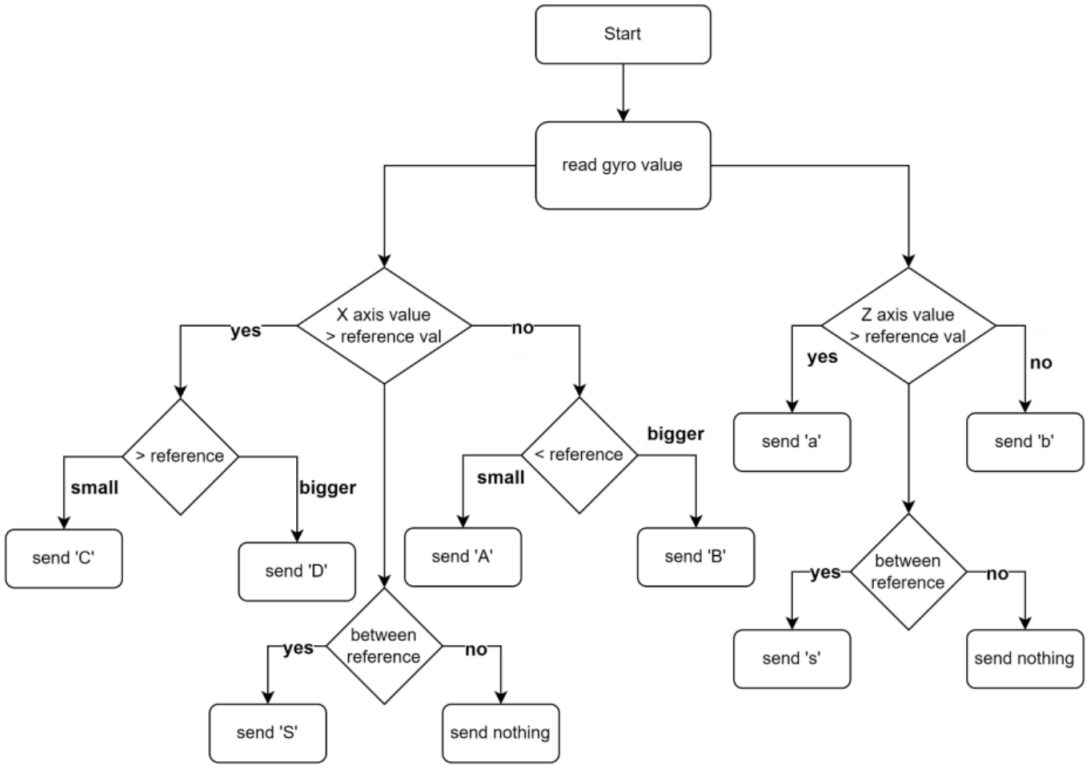


III. Algorithm

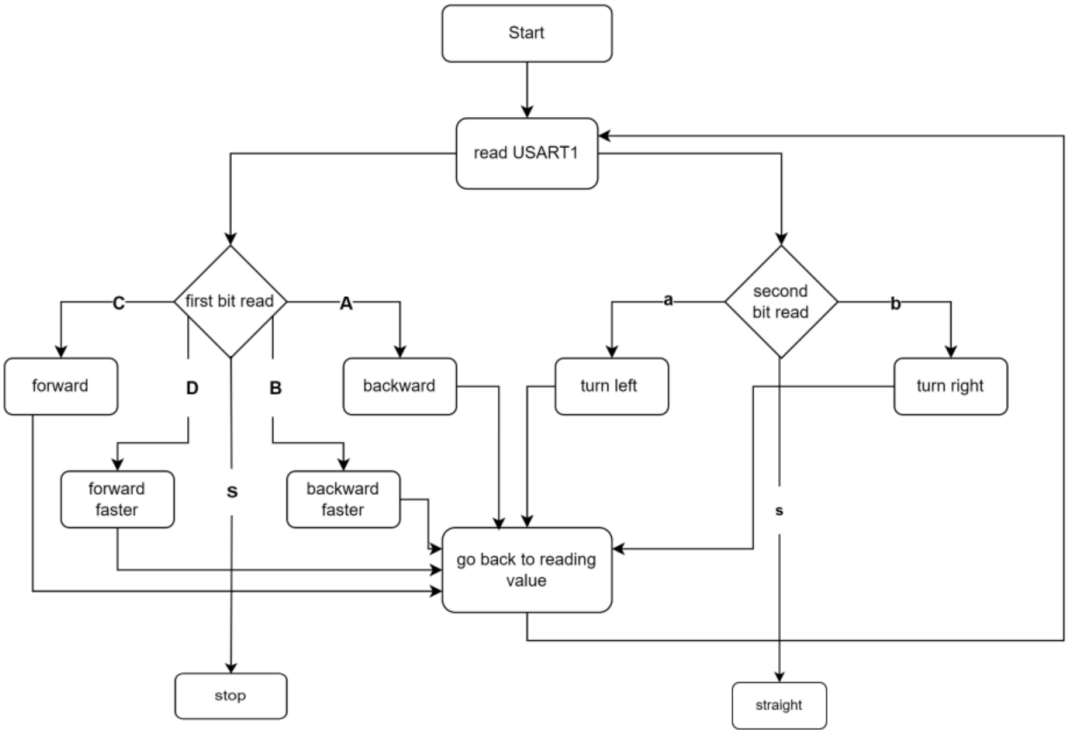
Logic Design

The first flow chart is for arduino sending the received data from the gyro sensor and processing the value in to a char and send to STM MCU attached to the RC car. It reads the X and Z value from the gyro then the value will be processed by reference value which was selected by several experiments. For X axis value if it is angled to the front it will send 'C' for more bigger angle to the front it will send 'D'. Oppositely, if it is angled to the back it will send 'A' and bigger angle to the back will send 'B'. If the value is neither both side it will send 'S'.

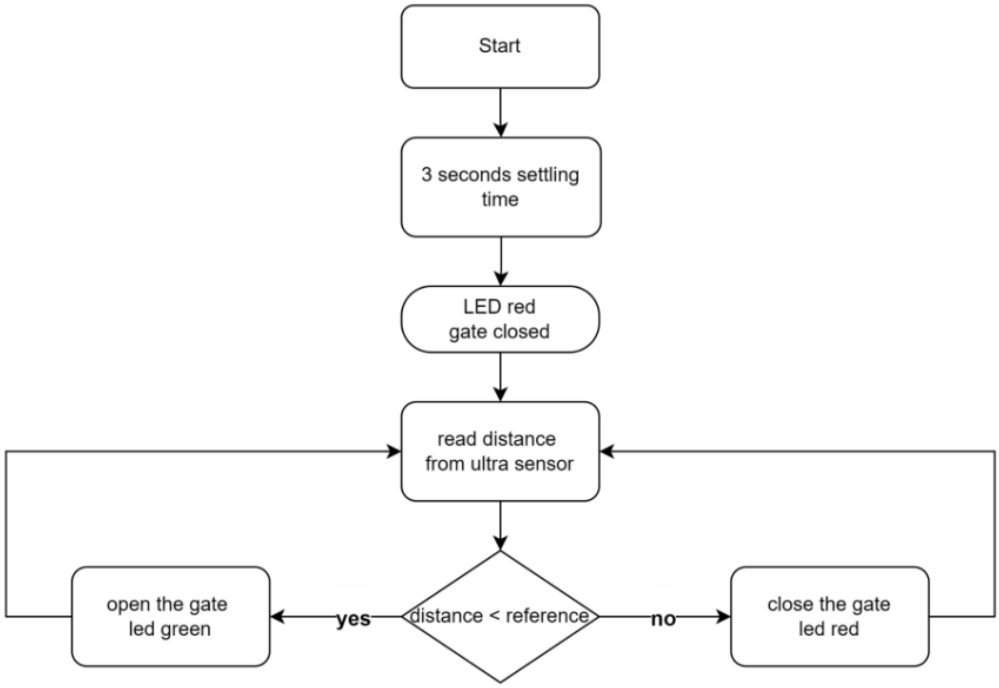
For the Z axis value if it is tilled to the left, it will send 'a' if tilled to the right it will send 'b'. If the value is neither both side it will send 's'. The character will be sent as two bits first will be the X axis valued and the following will be the Z axis value.



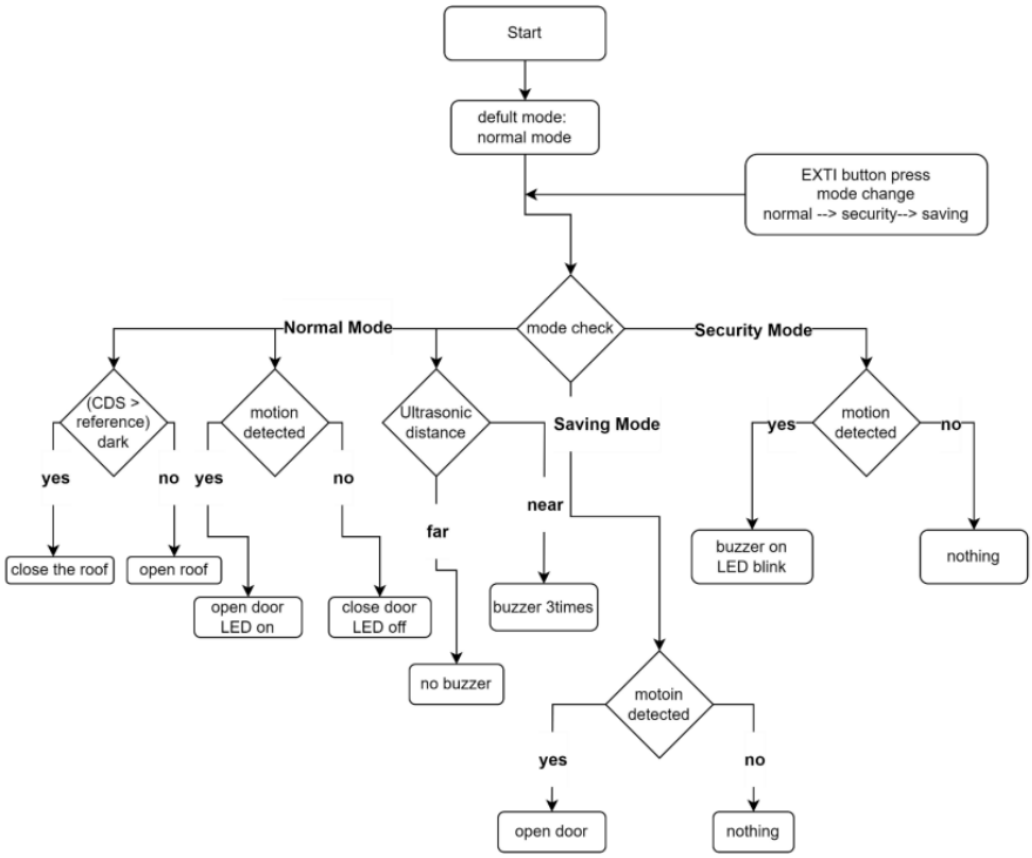
The second flow chart below shows the RC car's algorithm. First it reads the data sent by the arduino by USART1. 2 bits will be read and the first will concern with the DC motor and the second with the servo motor. If the first data is 'C' it goes forward by making the DC motor's direction and duty forwarded. And for 'D' it will raise the speed of the motor. If the received data is 'A' it go backward and 'B' for speed up. Last if it is 'S' it will stop. For the second received bit, if it is 'a' it will turn left by steps, 'b' turn right by steps, and 's' it will go straight.



The third flow chart if for the gate. First there is a settling time for 3 seconds. Then as default the LED and the gate is set as 'red on' and 'closed' respectively. If the ultrasonic sensor's value is under a reference value the gate will open and turn on the green LED. If not it will continuously turn the red LED and keep the gate closed.



The last flow chart is for the smart house. There is three mode for the house: normal, secure, and power-saving. Each button press by external interrupt will change the mode. If it is normal mode, the parking lot's roof will be closed when it is dark and opened when it is bright. If the motion is detected the door will open and LED will be on for a while. Also if the RC car get into the parking lot the buzzer will signal with 3 beeps. If it is in security mode, if the motion is detected the roof and the door will be closed and buzzer and LED will be turned on. If it is in saving mode, it will use only power that is necessary. The roof will always be closed and no buzz will signal and no LED will be turned on only the motion sensor will detect and open the door.



Code

- **MCU1(Parking Lot & Smart House)**

  Use several sensors to adjust the door, LED, roof, etc. of the house.

```
//////////////////// Normal Mode ////////////////////
        if(button_press == 0){

            // Buzz toggele
            if(distance < 15 && buz == 0){
                for(int i = 0; i < 5; i++){
                state = GPIO_read(GPIOA, 8);
                GPIO_write(GPIOA, 8, !state);
                delay_ms(500);
                }
            buz = 1;
            }

            else
            GPIO_write(GPIOA, 8, LOW);  // Buzz off

            // trash value
            if(distance > 90 && distance < 400)
            buz = 0;

            // Roof control by CDS sensor value
```

```
            if(value <= 1300)
                 count -= 1;
            else if(value > 1300)
                 count += 1;

            if(count >= 10) count = 9;
            else if(count <= 0) count = 1;

            PWM_duty(PWM_P, (0.5 + (1.f/9.f)*(float)count)/20.f);       // roof
duty

            // Motion detection sensor
            move = GPIO_read(GPIOA, 6);

            if(move){
                count2 = 9;
                GPIO_write(GPIOA, 9, HIGH); // if moving detection sensor
detected, LED ON
                PWM_duty(PWM_P2, (0.5 + (1.f/9.f)*(float)count2)/20.f); // Open
the DOOR
                delay_ms(3000);
            }

            GPIO_write(GPIOA, 9 ,LOW);  // LED OFF
            count2 -= 0.5;
            PWM_duty(PWM_P2, (0.5 + (1.f/9.f)*(float)count2)/20.f); // Closed
the DOOR

            if(count2 < 3) count2 = 3;

            printf("light = %f\r\n", value);
            delay_ms(500);
            }
```

The code above is a main code for smart house when it is in normal mode. If the value of 'button press' is 0 meaning no button press, automatically it will be in the normal mode. Buzzer will be worked by sensing the distance by ultrasonic sensor, roof will open and close by the value of 'count' which will be changed by CDS sensor value, and by motion sensor the door will be open and LED will be turned on or off.

```
//////////////////// Security Mode ////////////////////
        if(button_press == 1){  // safety mode
                move = GPIO_read(GPIOA, 6);
                if(move)
                    detect = 1;

                if(detect == 1){
                PWM_duty(PWM_P, (0.5 + (1.f/9.f)*(float)9)/20.f);       //
Closed the roof
                PWM_duty(PWM_P2, (0.5 + (1.f/9.f)*(float)3)/20.f);  // Closed
the DOOR
                state = GPIO_read(GPIOA, 8);
                for(int i = 0; i < 5; i++){
                    GPIO_write(GPIOA, 8, !state);   // BUZZ toggle

                LED_toggle();
                }
```

```
                delay_ms(500);
                }
        }
```

This is the code for security mode. If the button is pressed once the value of 'button_press' will be updated as 1, trigger to work security mode. Only the motion sensor will be considered in this mode. When the motion is detected buzzer will be beeping and LED will blink and all the openings roof and door will be closed until the mode is changed.

```
////////////////////// Saving Mode ////////////////////
        if(button_press == 2){
            move = GPIO_read(GPIOA, 6);
            if(move){
            count2 = 9;
            GPIO_write(GPIOA, 9, LOW);  // if moving detection sensor detected,
LED OFF
            PWM_duty(PWM_P2, (0.5 + (1.f/9.f)*(float)count2)/20.f); // Open the
DOOR
            delay_ms(3000);
            }
            count2 -= 0.5;
            PWM_duty(PWM_P2, (0.5 + (1.f/9.f)*(float)count2)/20.f); // Closed
the DOOR

            if(count2 < 3) count2 = 3;
        }
```

The last mode programmed in the main code is saving mode. Saving mode ignore all unnecessary power used. Only the motion sensor will be activate and when the motion is detected only the door will be opened not the LED and buzzer and roof.

```
void EXTI15_10_IRQHandler(void)
{
      time++;

      if (is_pending_EXTI(BUTTON_PIN) && time > 100000) {
                button_press ++;
      if (button_press > 3) button_press = 0;
      clear_pending_EXTI(BUTTON_PIN);
      }
}
```

The button interrupt is an external interrupt and EXTI handler is used. The role of variable 'time' is for avoiding bouncing problems of the physical button (debouncing method).

- **MCU2(Entrance)**

  Various sensors are used to adjust the door and indicate the status with LEDs.

```
int main(void){

   setup();
   printf("start\r\n");
   delay_ms(3000);
```

```
   while(1){
      PWM_duty(PWM_PIN, (0.5 + (1.f/9.f)*(float)count)/20.f);   // Control Servo
motor
      distance = (float) timeInterval * 340.0 / 2.0 / 10.0;    // [mm] -> [cm]

      if(distance > 2 && distance < 400){
         printf("distance = %f\r\n", distance);
         delay_ms(3000);
        // Control LED and Servo motor by distance
        if(distance < 15){
            count = 10;
            GPIO_write(GPIOA, 5, LOW);
            GPIO_write(GPIOA, 6, HIGH);
            PWM_duty(PWM_PIN, (0.5 + (1.f/9.f)*(float)count)/20.f);
            delay_ms(1000);
            }

         else if (distance >= 15){
            count = 0;
            GPIO_write(GPIOA, 5, HIGH);
            GPIO_write(GPIOA, 6, LOW);
         }
      }
   }
}
```

The code above is for entrance. First there is a settling time of 3 seconds. Then the ultrasonic sensor will continuously give the distance. When the distance is close enough meaning the RC car is in front of the gate, the green LED will turned on and the gate will be opened else, red LED will be turned on and the gate will be closed.

- **MCU3(RC car)**

```
if(is_USART_RXNE(USART1)){
                time++;
            if(time > 1000){
                PWM_duty(Motor_A, Duty_A);
                PWM_duty(Motor_B, Duty_B);
                GPIO_write(GPIOC, 10, dir);
                GPIO_write(GPIOC, 12, dir);
                PWM_duty(SERVO_PIN, (0.5 + (1.f/9.f)*(float)count)/20.f);

                // read two bit data
                for( i = 0; i < 2; i++){
                btData[i]  = USART1_read();
                }
                time = 0;

                printf("servo: %c\r\n\n",btData[0]);
                printf("DC motor: %c\r\n",btData[1]);

                // data for servo motor
                switch (btData[0]) {
                case 'b':
```

```
                    count = count + 0.4;
                    if (count > 26)
                        count = 26;                 // maximum value: 44
                    printf("count(b): %1.f\r\n", count);
                    break;
                case 'a':
                    count = count - 0.4;
                    if (count < 12)
                        count = 12;             // minimum value: 8
                    printf("count(a): %1.f\r\n", count);
                    break;
                case 's' || 'S':
                    count = 19;
                    break;
                }

                // data for DC motor
                switch (btData[1]) {
                case 'A':   //backward
                    dir = 0;
                    Duty_A = 0.5;
                    Duty_B = 0.5;
                    printf("backward\r\n");
                    break;
                case 'B':      //Backward Speed Up
                    dir = 0;
                    Duty_A = 0.9;
                    Duty_B = 0.9;
                    printf("BACK\r\n");
                    break;
                case 'C':   //forward
                    dir = 1;
                    Duty_A = 0.5;
                    Duty_B = 0.5;
                    printf("forward\r\n");
                    break;
                case 'D':   // Forward Speed Up
                    dir = 1;
                    Duty_A = 0.1;
                    Duty_B = 0.1;
                    printf("FORWARD\r\n");
                    break;
                case 'S':
                    dir = 0;
                    Duty_A = 0.0;
                    Duty_B = 0.0;
                    break;

                }
            }
        }
```

This code is in USART1_IRQHandler in RC car. The reason why this switch cases are in the USART handler is because the speed the arduino gives the data was too fast that it was impossible to run in the main code. Each time the arduino gives two bits of informations or in other words data so, the data is stored in btData array which has a size of 2. Capitalized characters A, B,C,D and S will

concern with DC motors and small letters a, b, and s will concern with the servo motor connected with steering.

- **Arduino**

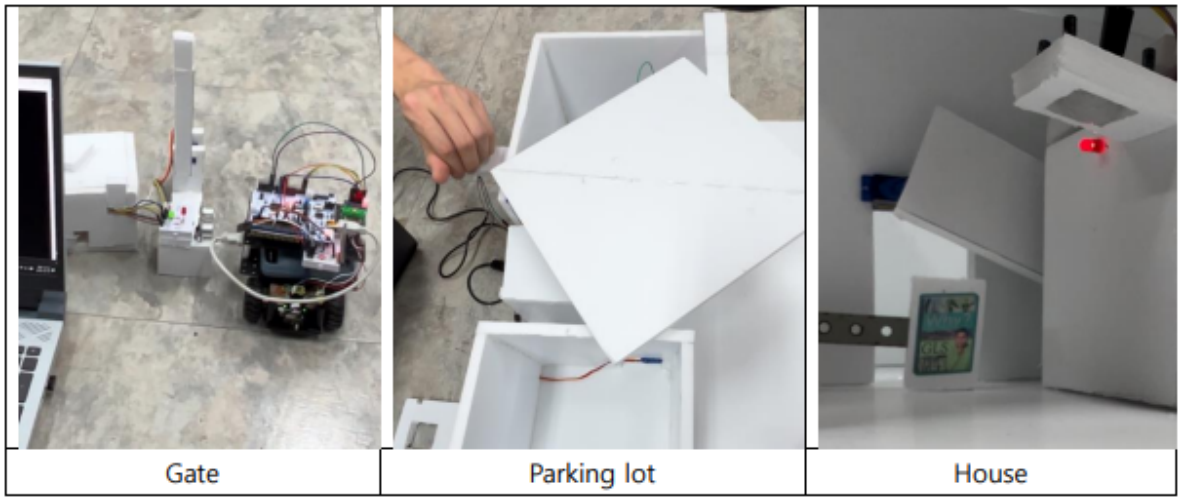
  The values of the gyro sensor are changed to alphabets and transmitted to the MCU board through Bluetooth communication.

```
  void loop() {
    mpu.update();

    HC06.write(val_X);  // send Alphabet of X angle to MCU
    HC06.write(val_Z);  // send Alphabet of Z angle to MCU

    if((millis()-timer)>10){ // print data every 100ms
    Serial.print("X : ");
    Serial.print(mpu.getAngleX());
    //Serial.print(val_X);
    Serial.print("\tZ : ");
    Serial.println(mpu.getAngleZ());
    //Serial.println(val_Z);
    timer = millis();
    }

    // DC Motor
    if(mpu.getAngleX()>=10 && mpu.getAngleX()<=30)  val_X = 'A';    // Go Back
    else if(mpu.getAngleX() > 30)  val_X = 'B';      // Speed up
    else if(mpu.getAngleX()<10 && mpu.getAngleX()>-10) val_X = 'S'; // Stop
    else if(mpu.getAngleX()<=-10 && mpu.getAngleX() >= -30) val_X = 'C';   // Go
straight
    else if(mpu.getAngleX()<-30) val_X = 'D';   // Speed up

    // USE Z
     if(mpu.getAngleZ()<=20 && mpu.getAngleZ()>=-20)  val_Z = 's';  // Stop
    else if(mpu.getAngleZ()>20)  val_Z = 'a'; // Left
    else if(mpu.getAngleZ()<-20)   val_Z = 'b';  // Right
  }
```

See Appendix for more information on the code

IV. Results and Demo

Images



Gate	Parking lot	House

Results

Developed using Arduino and MCU communications, the system uses various sensors to create smart parking and house automation solutions. It seamlessly controls vehicles, parking facilities, and house automation with motion detection, proximity sensors, gyro sensors, and light sensors. With Arduino and MCU communications, these components can be adjusted to perform certain actions based on predefined conditions or sensor inputs.

First, the values of the gyro sensor are transmitted to the vehicle through Bluetooth communication for control. The direction of the vehicle can be adjusted based on the angle of the gyro sensor, and the speed can also be adjusted.

second, when the vehicle enters the parking lot entrance, the vehicle is detected and the blocking door is opened. At this time, the LED changes from red to blue. Red means when the door is closed, and blue means when the door is open.

Third, when the vehicle approaches the parking lot, a buzzer sounds. Afterwards, when movement is detected by the motion detection sensor between the parking lot and the house, the LED in front of the house turns on and the house door opens automatically. Of course, it closes automatically after a certain time, and the LED turns off. This assumes that after parking the vehicle, a person gets out and enters the house. In addition, the parking lot roof opens when it is bright outside and closes when it is dark.

Next, up to the top, the mode of the parking lot is normal mode. The default mode is the normal mode. If we press the button on MCU_1 once, it changes to Security mode. In security mode, if the motion sensor detects movement, the garage roof and house door will close, the LED will flash, and the buzzer will sound continuously. At this time, the roof status does not change according to the CDS sensor and remains closed.

Finally, If you press the button on MCU_1 once more in Security mode, it switches to Saving mode. The goal in this mode is to reduce unnecessary energy consumption. Therefore, the buzzer does not make a sound even if it is a certain distance from the ultra sensor in the parking lot. Additionally, when the motion sensor detects movement, the house door opens automatically, but the LED does not turn on. As in normal mode, open doors will close after a certain period. In the case of the roof, it remains closed regardless of the CDS sensor. In this state, if you press the button on MCU_1 again, it changes back to Normal mode. Normal mode, Security mode, and Saving mode are repeated by the button.

Demo video: Link

V. Reference

Young-Keun Kim (2023): Link

Troubleshooting

The gyroscope sensor inherently includes a slight bias value. When calculating the actual angle using integration, the problem arises from accumulating this bias value, leading to divergence in the calculated results over time. I observed that reducing the delay accelerates the accumulation of this bias, resulting in faster divergence. Therefore, I mitigated the divergence rate by shortening the delay.

The servo motor lines were disconnected, so the motor did not operate normally. On the surface, I don't know well, but when I looked closely, there was a part where the line was disconnected. Therefore, the motor was replaced and used. And the angle that can be adjusted from RC car to servo motor is structurally limited. However, if the angle is given more than that in the code, the servo motor moves to the limit and can no longer move. But since the code is executed, the force is continuously applied to the servo motor and heated. In addition, a glue gun was used to attach the servo motor. However, due to the heating of the motor, the glue gun melted and the servo motor was separated. So, the problem was solved by changing the angle little by little finding the maximum angle, and applying it to the code.

Appendix

- **GetAngle (Arduino)**

```
#include <SoftwareSerial.h>
#include "Wire.h"
#include <MPU6050_light.h>

MPU6050 mpu(Wire);
unsigned long timer = 0;

SoftwareSerial HC06(8,9);

void setup() {
  Serial.begin(9600);
  Wire.begin();

  pinMode(2, INPUT);          // MPU6050 센서 INT핀모드 설정
  HC06.begin(9600);

  byte status = mpu.begin();
  Serial.print(F("MPU6050 status: "));
  Serial.println(status);
  while(status!=0){ } // stop everything if could not connect to MPU6050

  Serial.println(F("Calculating offsets, do not move MPU6050"));
  delay(500);
```

```
    mpu.calcOffsets(); // gyro and accelero
  }

  char val_X = 'S';
  char val_Z = 's';

  void loop() {
    mpu.update();

    HC06.write(val_X);  // send Alphabet of X angle to MCU
    HC06.write(val_Z);  // send Alphabet of Z angle to MCU

    if((millis()-timer)>10){ // print data every 100ms
    Serial.print("X : ");
    Serial.print(mpu.getAngleX());
    //Serial.print(val_X);
    Serial.print("\tZ : ");
    Serial.println(mpu.getAngleZ());
    //Serial.println(val_Z);
    timer = millis();
    }

    // DC Motor
    if(mpu.getAngleX()>=10 && mpu.getAngleX()<=30)  val_X = 'A';    // Go Back
    else if(mpu.getAngleX() > 30)  val_X = 'B';      // Speed up
    else if(mpu.getAngleX()<10 && mpu.getAngleX()>-10) val_X = 'S'; // Stop
    else if(mpu.getAngleX()<=-10 && mpu.getAngleX() >= -30) val_X = 'C';   // Go
straight
    else if(mpu.getAngleX()<-30) val_X = 'D';   // Speed up

    // USE Z
     if(mpu.getAngleZ()<=20 && mpu.getAngleZ()>=-20)  val_Z = 's';  // Stop
    else if(mpu.getAngleZ()>20)  val_Z = 'a'; // Left
    else if(mpu.getAngleZ()<-20)   val_Z = 'b';  // Right

  }
```

- **Parking_lot.c**

```
#include "ecSTM32F411.h"
#define PWM_P  PA_1 // roof
#define PWM_P2 PA_0 // door

#define TRIG PC_7
#define ECHO PB_6

void setup(void);

int buz = 0;
int state = 0;
int move = 0;   // moving detection sensor
float value = 0;    // jodo sensor value
static volatile float count = 9;    // roof angle control
static volatile float count2 = 3;   // door angle control
```

```c
int button_press = 0;
int button_state = 1;
int detect = 0;
int mode_change  = 3;

uint32_t ovf_cnt = 0;
float distance = 40;
float timeInterval = 0;
float time1 = 0;
float time2 = 0;
int time = 0;


int main(void){
    setup();
    printf("MCU Initialized\r\n");
    printf("light = %f", value);

    while(1){

        distance = (float) timeInterval * 340.0 / 2.0 / 10.0;    // [mm] -> [cm]
        printf("distance: %f\r\n", distance);
        //printf("buz: %d\r\n",buz);
        printf("mode: %d\r\n", button_press);

        ////////////////////// Normal Mode ////////////////////
        if(button_press == 0){
        // Buzz toggele
        if(distance < 15 && buz == 0){
            for(int i = 0; i < 5; i++){
            state = GPIO_read(GPIOA, 8);
            GPIO_write(GPIOA, 8, !state);
            delay_ms(500);
            }
            buz = 1;
        }
        else
            GPIO_write(GPIOA, 8, LOW);  // Buzz off

        // trash value
        if(distance > 90 && distance < 400)
            buz = 0;

        // Roof control by CDS sensor value
        if(value <= 1300)
                count -= 1;
     else if(value > 1300)
                count += 1;

     if(count >= 10) count = 9;
     else if(count <= 0) count = 1;

        PWM_duty(PWM_P, (0.5 + (1.f/9.f)*(float)count)/20.f);       // roof duty

      // Moving detection sensor
      move = GPIO_read(GPIOA, 6);
        if(move){
            count2 = 9;
```

```
            GPIO_write(GPIOA, 9, HIGH); // if moving detection sensor detected,
LED ON
            PWM_duty(PWM_P2, (0.5 + (1.f/9.f)*(float)count2)/20.f); // Open the
DOOR
            delay_ms(3000);
        }

        GPIO_write(GPIOA, 9 ,LOW);  // LED OFF
        count2 -= 0.5;
        PWM_duty(PWM_P2, (0.5 + (1.f/9.f)*(float)count2)/20.f); // Closed the
DOOR

        if(count2 < 3) count2 = 3;

        printf("light = %f\r\n", value);
        delay_ms(500);
        }

                ////////////////////// Security Mode ///////////////////
        if(button_press == 1){  // safety mode
                move = GPIO_read(GPIOA, 6);
                if(move)
                    detect = 1;

                if(detect == 1){
                PWM_duty(PWM_P, (0.5 + (1.f/9.f)*(float)9)/20.f);       //
Closed the roof
                PWM_duty(PWM_P2, (0.5 + (1.f/9.f)*(float)3)/20.f);  // Closed
the DOOR
                state = GPIO_read(GPIOA, 8);
                for(int i = 0; i < 5; i++){
                    GPIO_write(GPIOA, 8, !state);   // BUZZ toggle

                LED_toggle();
                }
                delay_ms(500);
                }
        }

                ////////////////////// Saving Mode ///////////////////
        if(button_press == 2){
            move = GPIO_read(GPIOA, 6);
            if(move){
            count2 = 9;
            GPIO_write(GPIOA, 9, LOW);  // if moving detection sensor detected,
LED OFF
            PWM_duty(PWM_P2, (0.5 + (1.f/9.f)*(float)count2)/20.f); // Open the
DOOR
            delay_ms(3000);
            }
            count2 -= 0.5;
        PWM_duty(PWM_P2, (0.5 + (1.f/9.f)*(float)count2)/20.f); // Closed the
DOOR

        if(count2 < 3) count2 = 3;
        }
    }
}
```

```
void setup(){
    RCC_PLL_init();
    SysTick_init();
    UART2_init();
    EXTI_init(GPIOC, BUTTON_PIN, FALL, 0);
    // input pin
    GPIO_init(GPIOA, 6, INPUT);
    GPIO_pupd(GPIOA, 6, EC_PD);
    GPIO_otype(GPIOA, 6, EC_PUSH_PULL);

    // led pin
    GPIO_init(GPIOA, 9, OUTPUT);
    GPIO_pupd(GPIOA, 9, EC_PD);
    GPIO_otype(GPIOA, 9, EC_PUSH_PULL);

    // buzz
    GPIO_init(GPIOA, 8, OUTPUT);
    GPIO_pupd(GPIOA, 8, EC_PD);
    GPIO_otype(GPIOA, 8, EC_PUSH_PULL);

    // button
    //GPIO_init(GPIOA, 3, INPUT);
    //GPIO_pupd(GPIOA, 3, EC_PD);
    //GPIO_otype(GPIOA, 3, EC_PUSH_PULL);
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);
    GPIO_pupd(GPIOC, BUTTON_PIN, EC_PD);
    GPIO_otype(GPIOC, BUTTON_PIN, EC_PUSH_PULL);


     // ADC Init
   ADC_init(PB_0);   // priority 1

     // servo motor1
     PWM_init(PWM_P);
   PWM_period_ms(PWM_P, 20);   // 20 msec PWM period

     // servo motor2
     PWM_init(PWM_P2);
   PWM_period_ms(PWM_P, 20);   // 20 msec PWM period

     // PWM configuration ---------------------------------------------------
---------------
   PWM_init(TRIG);         // PA_7: Ultrasonic trig pulse
   PWM_period_us(TRIG, 50000);     // PWM of 50ms period. Use period_us()
   PWM_pulsewidth_us(TRIG, 10);   // PWM pulse width of 10us

     // Input Capture configuration -----------------------------------------
--------------------------
   ICAP_init(ECHO);       // PB_6 as input caputre CH1
   ICAP_counter_us(ECHO, 10);      // ICAP counter step time as 10us
   ICAP_setup(ECHO, 1, IC_RISE);  // TIM4_CH1 as IC1 , rising edge detect
   ICAP_setup(ECHO, 2, IC_FALL);  // TIM4_CH2 as IC2 , falling edge detect
}

// Jodo Sensor
void ADC_IRQHandler(void){
```

```
   if(is_ADC_OVR())
      clear_ADC_OVR();

   if(is_ADC_EOC()){      // after finishing sequence
       value = ADC_read();
        }

}

// Ultra Sensor
void TIM4_IRQHandler(void){
   if(is_UIF(TIM4)){                        // Update interrupt
      ovf_cnt++;                           // overflow count
      clear_UIF(TIM4);                             // clear update interrupt
flag
 }
   if(is_CCIF(TIM4, 1)){                          // TIM4_Ch1 (IC1) Capture Flag.
Rising Edge Detect
      time1 = TIM4->CCR1;                           // Capture TimeStart
      clear_CCIF(TIM4, 1);                 // clear capture/compare interrupt
flag
   }
   else if(is_CCIF(TIM4, 2)){                    // TIM4_Ch2 (IC2) Capture Flag.
Falling Edge Detect
      time2 = TIM4->CCR2;                           // Capture TimeEnd
      timeInterval = ((time2-time1)+(TIM4->ARR+1)*ovf_cnt)/100;    // (10us *
counter pulse -> [msec] unit) Total time of echo pulse
      ovf_cnt = 0;                          // overflow reset
      clear_CCIF(TIM4,2);                           // clear capture/compare
interrupt flag
   }
}

void EXTI15_10_IRQHandler(void)
{
      time++;

      if (is_pending_EXTI(BUTTON_PIN) && time > 100000) {
                button_press ++;
      if (button_press > 3) button_press = 0;
      clear_pending_EXTI(BUTTON_PIN);
      }
}
```

- **Entrance.c**

```
# include "ecSTM32F411.h"

#define PWM_PIN PA_1

#define TRIG PC_7
#define ECHO PB_6

uint32_t ovf_cnt = 0;
float distance = 0;
float timeInterval = 0;
float time1 = 0;
```

```
float time2 = 0;

void setup(void);

static volatile float count = 0;

int main(void){

   setup();
   printf("start\r\n");
   delay_ms(8000);

   while(1){
      PWM_duty(PWM_PIN, (0.5 + (1.f/9.f)*(float)count)/20.f);   // Control Servo
motor
      distance = (float) timeInterval * 340.0 / 2.0 / 10.0;    // [mm] -> [cm]

      if(distance > 2 && distance < 400){
         printf("distance = %f\r\n", distance);
         delay_ms(1000);

                // Control LED and Servo motor by distance
         if(distance < 15){
            count = 10;
                     GPIO_write(GPIOA, 5, LOW);
                       GPIO_write(GPIOA, 6, HIGH);
                    PWM_duty(PWM_PIN, (0.5 + (1.f/9.f)*(float)count)/20.f);
                    delay_ms(1000);
                 }

         else if (distance >= 15){
            count = 0;
                     GPIO_write(GPIOA, 5, HIGH);
                       GPIO_write(GPIOA, 6, LOW);
                 }
      }
   }
}

void TIM4_IRQHandler(void){
   if(is_UIF(TIM4)){                         // Update interrupt
      ovf_cnt++;                            // overflow count
      clear_UIF(TIM4);                                 // clear update interrupt
flag
 }
   if(is_CCIF(TIM4, 1)){                          // TIM4_Ch1 (IC1) Capture Flag.
Rising Edge Detect
      time1 = TIM4->CCR1;                              // Capture TimeStart
      clear_CCIF(TIM4, 1);                // clear capture/compare interrupt
flag
   }
   else if(is_CCIF(TIM4, 2)){                    // TIM4_Ch2 (IC2) Capture Flag.
Falling Edge Detect
      time2 = TIM4->CCR2;                              // Capture TimeEnd
      timeInterval = ((time2-time1)+(TIM4->ARR+1)*ovf_cnt)/100;    // (10us *
counter pulse -> [msec] unit) Total time of echo pulse
      ovf_cnt = 0;                           // overflow reset
```

```
      clear_CCIF(TIM4,2);                           // clear capture/compare
interrupt flag
   }
}

void setup(void){

   RCC_PLL_init();
   SysTick_init();
   UART2_init();

   PWM_init(PWM_PIN);
   PWM_period_ms(PWM_PIN, 20);   // 20 msec PWM period

   EXTI_init(GPIOC, BUTTON_PIN, FALL, 0);
   GPIO_init(GPIOC, BUTTON_PIN, INPUT);
   GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU);

     GPIO_init(GPIOA, 5, OUTPUT);                          // LED RED
     GPIO_pupd(GPIOA, 5, EC_PD);
     GPIO_otype(GPIOA, 5, EC_PUSH_PULL);
     GPIO_init(GPIOA, 6, OUTPUT);                          // LED BLUE
     GPIO_pupd(GPIOA, 6, EC_PD);
     GPIO_otype(GPIOA, 6, EC_PUSH_PULL);

   SysTick_init();
   UART2_init();

     // PWM configuration ------------------------------------------------------
---------------
   PWM_init(TRIG);         // PA_7: Ultrasonic trig pulse
   PWM_period_us(TRIG, 50000);     // PWM of 50ms period. Use period_us()
   PWM_pulsewidth_us(TRIG, 10);   // PWM pulse width of 10us


     // Input Capture configuration --------------------------------------------
---------------------------
   ICAP_init(ECHO);       // PB_6 as input caputre CH1
   ICAP_counter_us(ECHO, 10);      // ICAP counter step time as 10us
   ICAP_setup(ECHO, 1, IC_RISE);  // TIM4_CH1 as IC1 , rising edge detect
   ICAP_setup(ECHO, 2, IC_FALL);  // TIM4_CH2 as IC2 , falling edge detect
}
```

- **ecSTM32F411.h**

```
#include "ecEXTI.h"
#include "ecGPIO.h"
#include "ecPinNames.h"
#include "ecPWM.h"
#include "ecRCC.h"
#include "ecSysTick.h"
#include "ecTIM.h"
#include "ecPWM.h"
#include "ecPinNames.h"
#include "ecStepper.h"
#include "ecUART.h"
#include "ecADC.h"
```

```
#include "stm32f411xe.h"
#include "stm32f4xx.h"
#include "math.h"
```