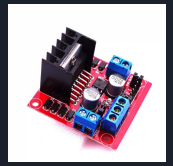
Mini Robot Cleaner

CHENG, Wing Ching 20713627
Young, James Yang 20740589
WONG, Ho Leong 20793677

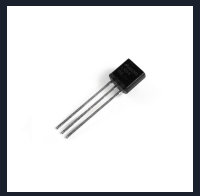
Overview

3 x Ultrasonic sensor - HC SR04

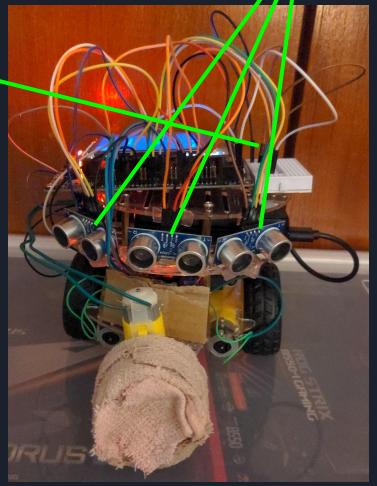


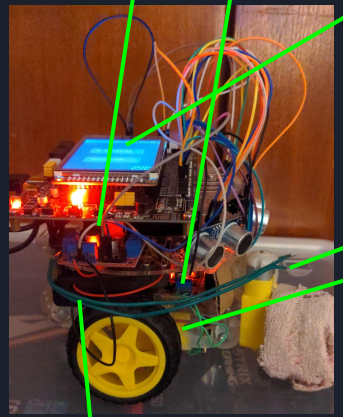
L298N Motor Driver x 2

LCD Display - ILI9314



LM35 Temperature Sensor





DC TT Motor x3

Battery x2
(Power Supply)

Overview

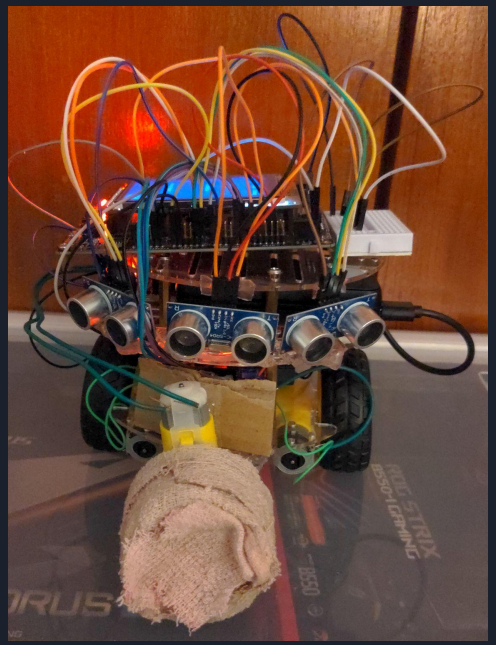
Modes (functions):

- Wireless manual control of robot

- Automatic mode with avoid obstacles while free roaming

- GUI on LCD display that shows robot information and allows users to change mode with touch screen

- Remote viewing of temperature data



LCD Display

- To create a modern looking GUI, used LVGL (popular free and open-source embedded graphics library)
- To port LVGL to STM32, set a few parameters (display size, colour, etc.), and implement a driver function for both touch screen and display
- To increase performance, we used DMA memory to memory to copy array of pixels into given area of display rather than drawing it pixel by pixel as shown in their documentation
- Also had to add optimization to compiler and remove unneeded components as LVGL unoptimized had size of ~ 500 kb, with optimization down to ~ 250 kb


```c
void my_flush_cb(lv_disp_drv_t * disp_drv, const lv_area_t * area, lv_color_t * color_p)
{
    /*The most simple case (but also the slowest) to put all pixels to the screen one-by-one
     * `put_px` is just an example, it needs to implemented by you.*/
    int32_t x, y;
    for(y = area->y1; y <= area->y2; y++) {
        for(x = area->x1; x <= area->x2; x++) {
            put_px(x, y, *color_p);
            color_p++;
        }
    }
```

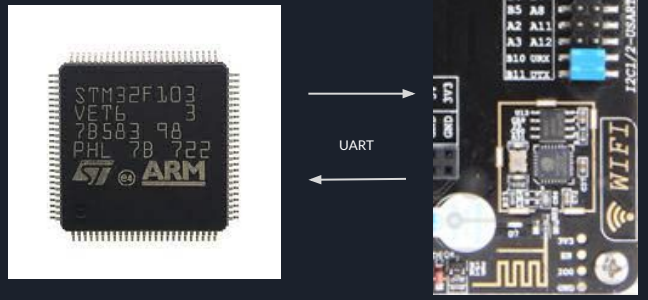
Inefficient display method

```c
// optimization attempt using dma
ILI9341_OpenWindow2(area->x1,area->y1,area->x2,area->y2);
uint32_t size = (area->x2 - area->x1 + 1) * (area->y2 - area->y1 + 1);
HAL_StatusTypeDef DMA_status = HAL_ERROR;
// start dma transfer, parameters: pointer to dma channel, data, destination buffer address, size of data, send to 0x60020000 (FSMC ILI9341 data address)
HAL_DMA_Start(&hdma_memtomem_dma1_channel1,(uint32_t)color_p,(uint32_t)0x60020000,size);
HAL_DMA_PollForTransfer(&hdma_memtomem_dma1_channel1, HAL_DMA_FULL_TRANSFER, 100);
```

More efficient display method

Wireless Control Mode Overview

- Use on-board ESP8266 for wireless functions
- STM32 communicates to ESP8266 through UART3 via jumper (eg. B10 UART3_TX -> ESP8266 UART_RX)
- STM32 sends series of AT commands to tell ESP8266 what to do
- Data received by ESP8266 stored in a buffer
- Check buffer to see what command is received for robot to do (eg. if buffer contains string /F-, this means to move robot forward)



ESP8266 IO	STM32 IO
URXD	PB10
UTXD	PB11
CH_PD	PB8
RST	PB9

From Schematic

Wireless Control Mode

- Mainly referenced ESPRESSIF AP commands documentation for what AT commands to send

- Example on the right of AT commands to create UDP server and the code implementation

- Our robot car connects to Wi-fi on startup, creates a UDP server on port 4545 and for sending data we initialize the IP address it will send to

- Wireless control and server to retrieve data written in Python and GUI created with PyQt5

ESP AT Commands Documentation

Establish UDP Transmission

Set Command

Command:

```
// Single connection (AT+CIPMUX=0):
AT+CIPSTART=<"type">,<"remote host">,<remote port>[,<local port>,<mode>,<"local IP">]

// Multiple connections (AT+CIPMUX=1):
AT+CIPSTART=<link ID>,<"type">,<"remote host">,<remote port>[,<local port>,<mode>,<"local IP">]
```

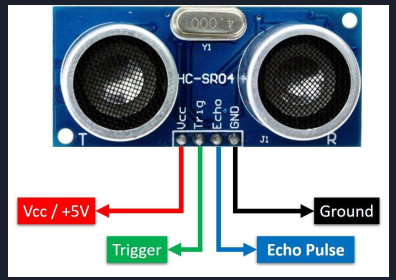
Code Implementation

```
// create udp server
void createUDPServer()
{
    // allow multiple connections
    sendData("AT+CIPMUX=1\r\n");
    HAL_Delay(1000);
    clearReceivedBuffer();
    // create udp server with own ip at port 4545
    sendData("AT+CIPSTART=0,\"UDP\",\"0.0.0.0\",4545,4545,2\r\n");
    HAL_Delay(3000);
    showResponse();
}
```

Desktop GUI for wireless control

Detection components - Ultrasonic Sensor

- HC-SR04 to detect the obstacle
- Use TIM to send PWM to ultrasonic sensor and return the data (distance)
- Range: 2cm – 90cm non-contact measurement data



Code Implementation

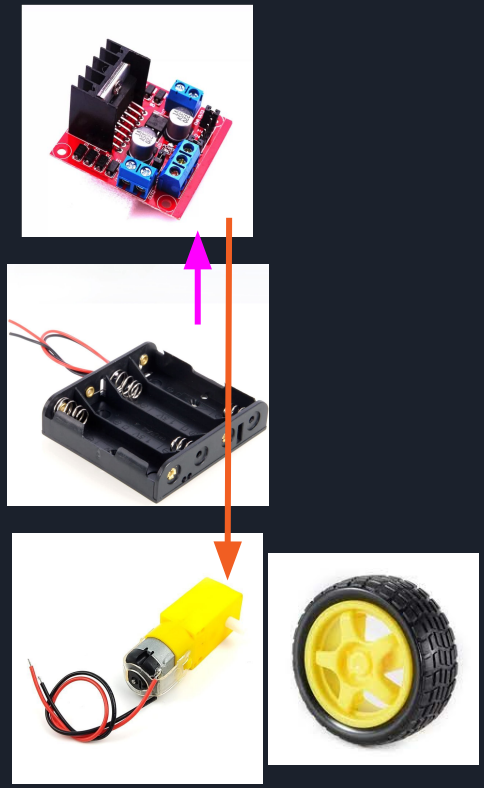
```
//Middle
/* Set TRIG On */
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
HAL_Delay(15);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
/* wait for echo signal is off */
while(HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_6) == GPIO_PIN_RESET);
/* measure time while high */
__HAL_TIM_SET_COUNTER(&htim3,0);
while(HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_6) == GPIO_PIN_SET);
uint16_t counter1 = __HAL_TIM_GET_COUNTER(&htim3);
/* calculate distance(cm) */
mid = counter1 * .034 / 2;
```

Robot Car Components

- Use GPIO to connect L298N Motor Driver to control the speed & direction by PWM

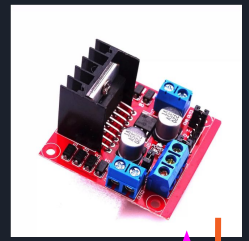
- The external 6V battery will be the power supply to Motor Driver

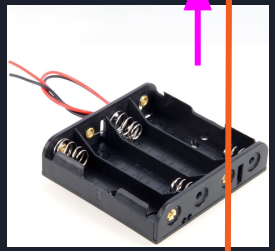
- Motor Driver power DC motor to move
  Use DC motor to move the car wheels



Cleaning Motor

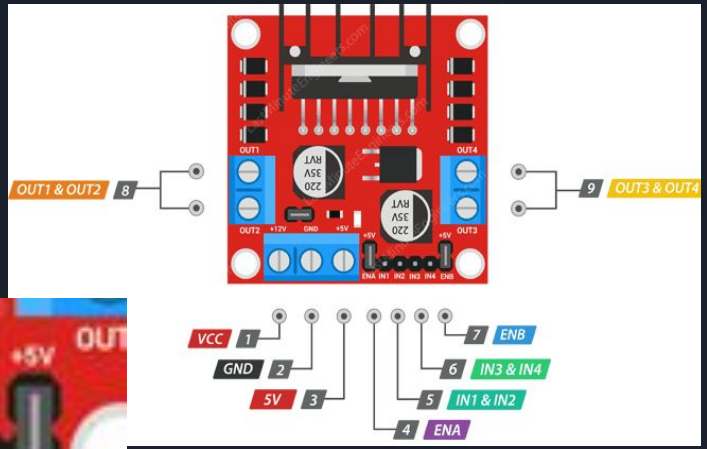
- Same as previous slide
- Towel stick to the motor wheel to achieve the mopping floor function
- On / Off option





Robot Car Components (L298N)

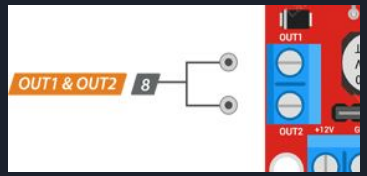
L298N overview



- IN1 & IN2: control **left** wheel direction / stop
- IN3 & IN4: control **right** wheel direction / stop

- ENA: control the speed of **left** wheel by PWM
- ENB: control the speed of **right** wheel by PWM

- OUT1 & OUT2: power to **left** DC motor
- OUT3 & OUT4: power to **right** DC motor



Pin Control

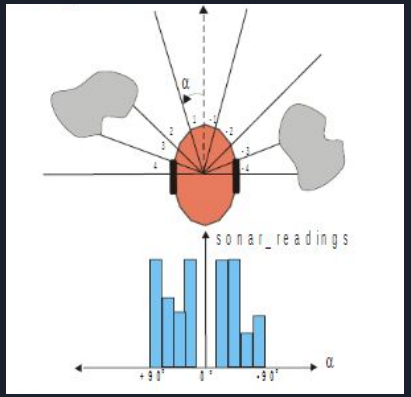
Input1	Input2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

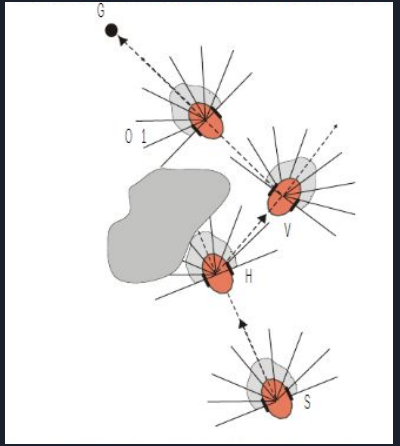
Auto Mode

- Automatically avoid obstacles

- The Bubble rebond Algorithm (real-time obstacle avoidance algorithm)

- Step 1: receive data from the 3 ultrasonic sensors
- Step 2: input data to the algorithm and calculate the ideal path
- Step 3: Control the motor to avoid obstacle





Temperature Sensor (LM35)

Useful features in this project:

- **Linear + 10-mV/ C Scale Factor**

→ Vout (mV) = 10 * ADC Value

- **Rated for Full -55 C to 150 C Range**

→ Can cover 150 C high temperature

- **Accuracy at 25 C is +0.5 to -0.5**

→ Acceptable range

TEXAS INSTRUMENTS

SNIS159H – AUGUST 1

LM35 Precision Centigrade Temperature Sensors

1 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates From 4 V to 30 V
- Less Than 60-μA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

3 Description

The LM35 series are pre temperature devices with an proportional to the Centigr LM35 device has an a temperature sensors calibrate is not required to subtract a from the output to obtain scaling. The LM35 device external calibration or trimm accuracies of ±¼°C at room over a full −55°C to 150°C te cost is assured by trimming wafer level. The low-output im and precise inherent calibrati makes interfacing to reado

PARAMETER	VALUE
Accuracy at 25°C	±0.5°C
Accuracy from –55 °C to 150°C	±1°C
Temperature Slope	10 mV/°C

Temperature Sensor (LM35)

For implementation:

ADC Poll for Conversion → Calculation → Swap the data type (for printing on LCD)

Why divide 4095?

The STM32 ADC has a resolution of 12-Bit which results in a total conversion time of Sampling Time+12.5 clock cycles.

12-bits = 4096

Start from 0 → 4096-1 = 4095

```
HAL_ADC_Start(&hadc1);
      if (HAL_ADC_PollForConversion(&hadc1,10) == HAL_OK)
      {
            adcvalue = HAL_ADC_GetValue(&hadc1);
            temp = (adcvalue*1.5*100)/4095.0;
            sprintf(MSG,"%.2f C",temp);
```

Limitations

- Using Primary Battery for the power supply → X rechargeable
- Lack of cooldown function → Overheat (Environment? Using time?)
- Limited angle of detecting obstacle (only X-Y axis)
- Lack of protection (waterproof , shield)

- Obstacle avoidance implementation still not optimal

→ Easy damaged

Other References

Documentation/Schematics:

- HR-S04 Documentation
- LM35 Documentation
- Mini V3 Schematics
- ILI9341 Documentation
- LVGL Documentation
- ESP8266 AT Commands Documentation
- L298N Documentation
- DC Gear TT Motor Documentation
- Real Time Avoid Algorithm Documentation

Hardware connections