EEEE1039
Applied Electrical and Electronic Engineering: Construction Project

Basics of Building and Testing the Vehicle, Real-Time (RT) Control and Human Machine Interface (HMI)

Department of Electrical and Electronic Engineering
University of Nottingham Ningbo China

Student's Name: Jingxiong Yu
Student's ID: 20123738
Group: 7
Tutors' Name: Dr. Chunyang Gu, Dr. Jing Wang,
Dr. Giampaolo Buticchi, Dr. Chiew-Foong Kwong
Submission Date: 22 November 2019

Abstract

The purpose for this experiment was to build a vehicle with Real-Time control and Human Machine Interface. Therefore, in this report, process of building the vehicle, some concerning theory and possible analysis were displayed. For theory, interfacing with Arduino GPIOs, powering up Arduino, real-time controlling and human machine interfacing techniques were presented. All data and graphs collected during the experiment were also covered. Circuits and codes were very important as this experiment was operated on the condition of many modifications and testing of circuits and codes. Human Machine Interface recognized commands from operators and made reflections accordingly, which were input values for the vehicle, after a data processing and reaction taken, an advanced vehicle with Real-Time control and Human Machine Interface was successfully built.

Introduction

Objectives

The objective of this experiment was to finally build a vehicle that was able to finish final tasks about spinning and running with real-time control and had a human machine interface for operators to input commands before reactions taken. The experiments were conducted to build the advanced vehicle. The usage of Arduino, theory of H-bridge, GPIO, MCU digital I/O, Real-Time communication and control, logic gates, PWM and HMI, techniques for soldering works, programming and debugging for both software and hardware for vehicle's challenge tasks were practiced during the process. With a full understanding of the codes and modification, proper modifying and testing, the advanced vehicle was successfully built and operated.

Some hardware was required to be integrated and assembled on the vehicle, including an Arduino, a robot baseboard, robot skeletons, wires, reducing motors 25GA-370 with encoder, soldered H-bridge PCBs, wheels, battery set with a case, stripboards and additional components for human machine interface and RT control which were 5mm LEDs, a WH148 potentiometer 10kOhm, transistors S8050 and S9014, a FR107 fast recovery diode, LM358 and LM386, an active buzzer, a passive buzzer, a LCD 1602 screen, an EC11 digital encoder with button module, a microphone 6*5mm, press button switches 6*6*7 and SS12D00G3 toggle switches.

Coding was also needed for driving the vehicle and the RT communication and control as well as the HMI parts, including the spinning and running straight tasks as well as the task, speed and target selection tasks. Real-Time communication and control methods were applied for the instant input and output to control and drive the vehicle, based on the MsTimer2 Interrupt Service Routine algorithm. Therefore, the frequency of the loop in which the Arduino sent the commands to the baseboard and received the feedback from the baseboard was artificially controlled to 200 Hz. The commands the Arduino recognized from HMI were input values for the vehicle, after a data processing via software, the vehicle was able to take reactions to certain commands.

Lab Session I:

1. Kept recording in lab logbooks.
2. Exercised soldering for SMD components individually and exercised for H-bridge together. Finished the tasks of "H bridge Project".
3. Got familiar with lab equipment including multi-meters, oscilloscopes, DC power sources. Passed the skill test individually.
4. Got familiar with Arduino and finished the tasks of "Arduino Introduction".
5. Finished the soldering and debugging of one H-bridge individually. Verified the performance with Arduino software.
6. Built the vehicle. Integrated all the components together and attempted to complete tasks for the week.

Lab Session II:

1. Studied about MCU digital I/O, interrupts, Real-Time communication and control.
2. Re-burned the baseboard with a given RT control firmware.

3. Designed and implemented RT communication between the baseboard and the Arduino Nano, and also the RT control.
4. Verified the performance (frequency >= 200Hz) using oscilloscope.
5. Studied logic gates and PWM. Recorded the waveforms of IRS2008 input/output (any channel), and the waveform of the associated motor voltage.
6. Studied, implemented and tested different input/output HMI parts with breadboard.
7. Designed the HMI to meet the requirement of demo, soldered and debugged the HMI on a stripboard.
8. Integrated the vehicle with the HMI board. Designed the code and debugged for the required challenge. Attempted to complete tasks for the week.

Backgrounds

Lab Session I:

The hardware parts used for the lab session were Skeleton Bot (4WD hercules mobile robotic platform), self-soldered H-bridge PCB for the driving of motors and control system of vehicle using Arduino. The hardware parts were shown in Figure 1 – 3.

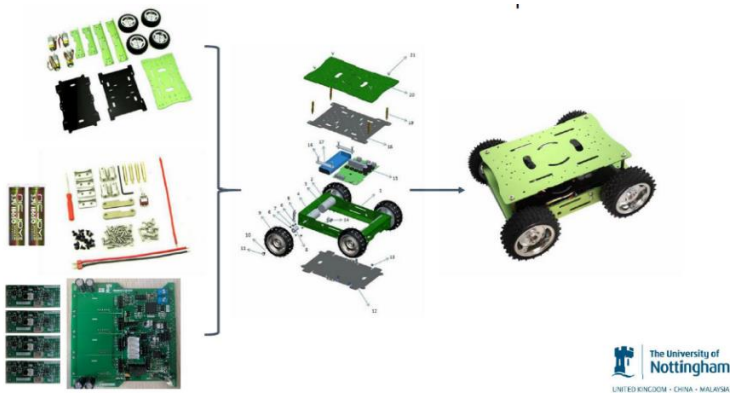


Figure 1. Skeleton Bot

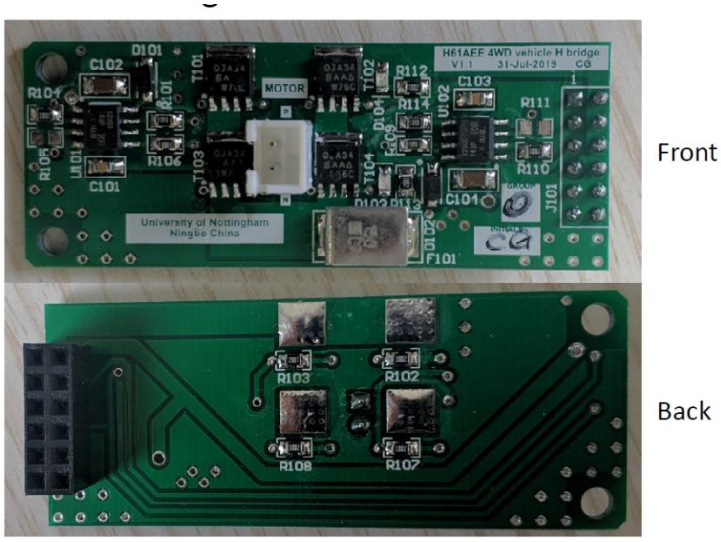


Figure 2. H-bridge PCB

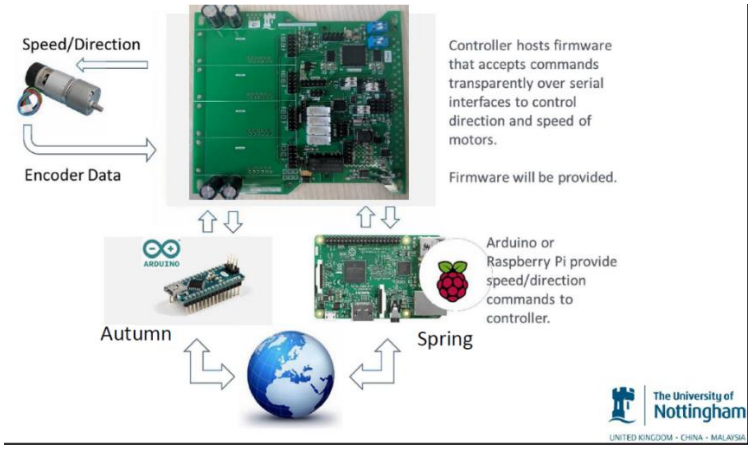


Figure 3. Control system of vehicle using Arduino

The software parts used for the lab session was 15A 6-20V Motor Controller for 4 motors using serial communication methods including SPI, I2C and UART.

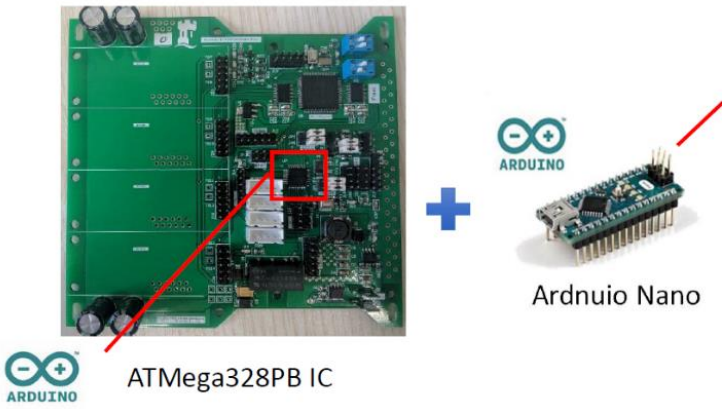


Figure 4. 15A 6-20V Motor Controller (4 motors)

About Arduino, different from C/C++, main function was embedded for Arduino as shown in Figure 5, therefore, two functions setup() and loop() were needed in Arduino sketches.

```
int main(void)
{
        init();

        setup();

        for (;;)
                loop();

        return 0;
}
```

Figure 5. Embedded main function for Arduino

Lab Session II:

The Arduino Nano board worked as a controller for the vehicle system, which was actually a typical control system. The two different kinds of Loop Control were shown in Figure 6.

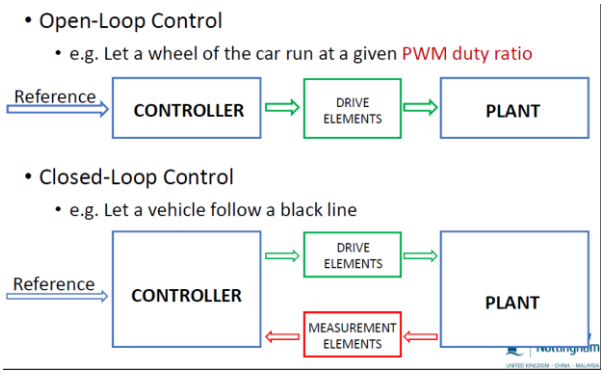


Figure 6. Loop Control

As shown in Figure 7, the ATMEGA328PB was the MCU used on Arduino Nano.

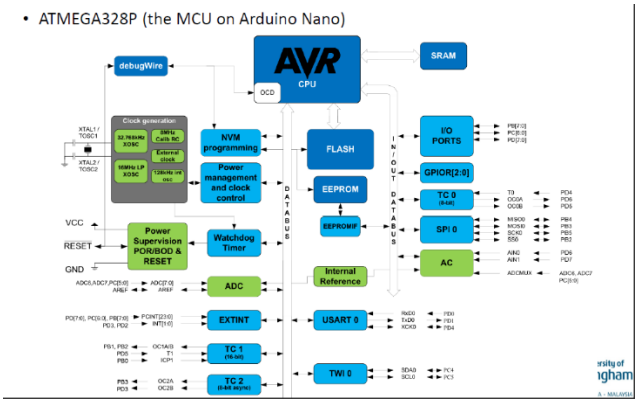


Figure 7. ATMEGA328PB

As shown in Figure 8, a human-machine interface (HMI) was the user interface that connected an operator to the controller for an industrial system.

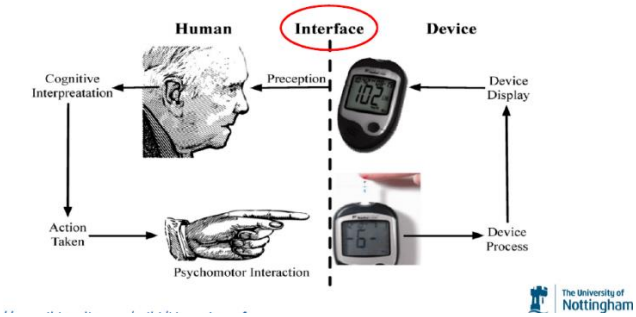


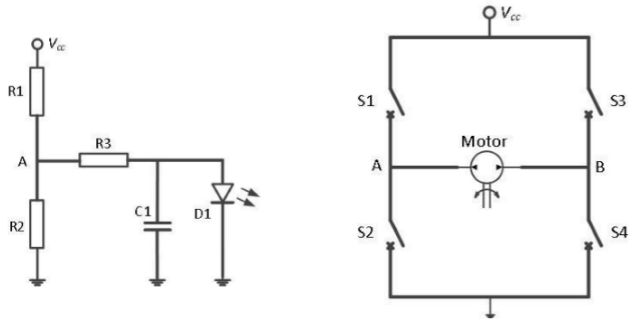
Figure 8. HMI

Methods

Lab Session I:

Task 1: Potential Divider and H-bridge exercises and tasks.

The circuits shown in Figure 9a presented a potential divider with a LED of which a variation was used later in the project. The circuit shown in Figure 9b was an H-bridge motor controller on which the main development of the project was conducted.



a) The LED circuit. b) The motor controller.

Figure 9. The LED circuit and H-bridge circuit

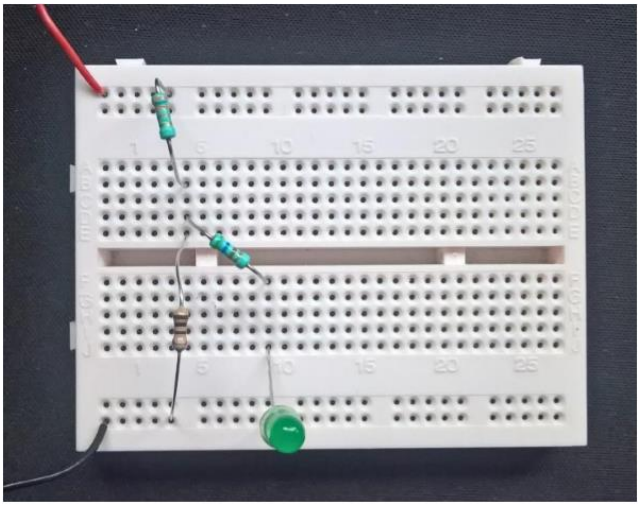


Figure 10. The LED circuit used for task

The circuit shown in Figure 10 was the one used for the following experiment. There were three attempts in this part of experiment. The detailed values of three resistors for each attempt were shown in Table 1.

Table 1. Detailed values of resistors

	R1	R2	R3
1st Attempt	30 Ω	10 Ω	1.5 kΩ
2nd Attempt	360 Ω	120 Ω	15 kΩ
3rd Attempt	360 Ω	120 Ω	1.5 kΩ

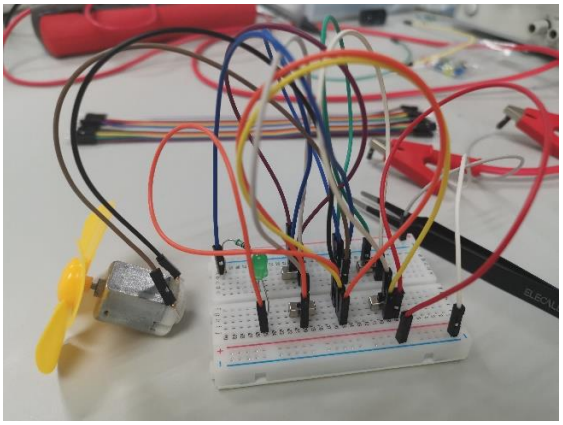


Figure 11. H-bridge on breadboard

The circuits shown in Figure 9b and Figure 11 were used for the H-bridge experiments. And the basic operation and theory of the H-bridge were shown in the Figure 12.

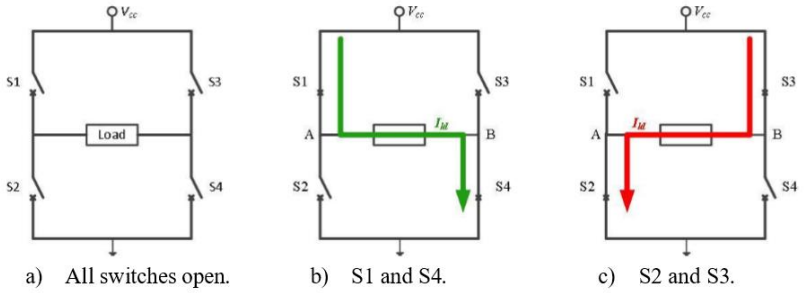


Figure 12. The H-bridge: Basic operation

In the first experiment, the basic operation of H-bridge was tested. Relating data for the experiments was shown in the Table 2.

Table 2. Basic operation of H-bridge experiment

	Voltage	S1	S2	S3	S4
1	1.5 V	OFF	OFF	OFF	OFF
2	1.5 V	ON	OFF	OFF	ON
3	1.5 V	OFF	ON	ON	OFF
4	3 V	OFF	OFF	OFF	OFF
5	3 V	ON	OFF	OFF	ON
6	3 V	OFF	ON	ON	OFF

In the second experiment, the fan on the motor was pressed by a finger to stop the motor, therefore, the electrical power instead of became mechanical power but became heat so that the current flowed through the circuit was different from the current in the previous experiment. The relating experimental environment and operations were the same as the previous experiment shown in Table 2.

In the third experiment, the current when the fan was about to rotate but not was measured in two different fan rotating directions clockwise and anti-clockwise.

Experiments shown above were some of the tasks picked from "H Bridge Project", other tasks were also finished following the instructions but were not displayed in this report due to the limitation of the length of the report.

Task 2: Got familiar with lab equipment and passed the skill test individually.

The basic operation of lab equipment including multi-meters, oscilloscopes and DC power sources was studied by group members together. And the skill test was attempt by each of the group members individually.

Task 3: Got familiar with Arduino and finished the tasks of "Arduino Introduction".

The circuit used for the "Blink" LED experiment was shown in Figure 13, and the Arduino was connected to PC via micro USB cable. And the code used was shown in Figure 14.

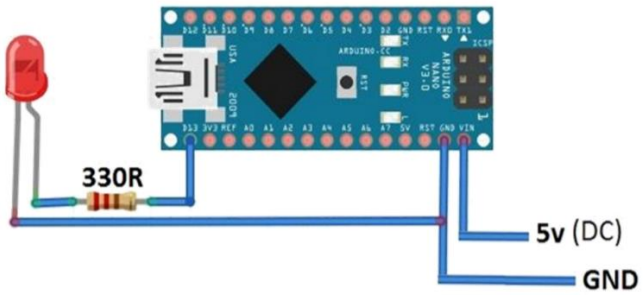


Figure 13. "Blink" LED circuit

```
// the setup function runs once when you press reset or power the board

void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever

void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Figure 14. Code for "Blink" LED

Experiments shown above were some of the tasks picked from “Arduino Introduction”, other tasks were also finished following the instructions but were not displayed in this report due to the limitation of the length of the report.

Task 4: Finished the soldering and debugging of one H-bridge individually. Verified the performance with Arduino software.

The 4 H-bridge PCBs were soldered and debugged individually by group members of group 7. And the verification of the H-bridge soldering work was attempted. Good and bad solder joints examples were shown in Figure 15.

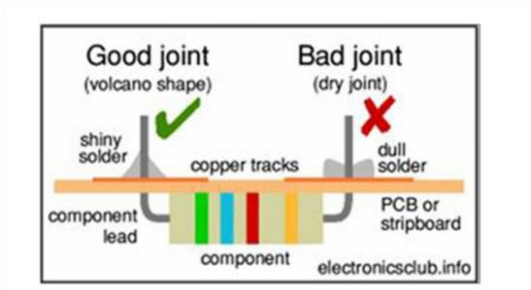


Figure 15. Good and bad solder joints

Task 5: Built the vehicle. Integrated all the components together and attempted to complete tasks for the week.

After the integration of all the components for the vehicle, the baseboard and motors were tested by sending commands via CH340 TTL to USB module driver with Serial Monitor: “#Baffff050,050,050,050”.

After the testing, the code for the demo tasks was programmed and debugged.

For this session, the UART serial communication method was applied to the vehicle. In theory, for the UART communication, three pins from the baseboard including GND, TX and RX were connected with the Arduino accordingly. But the program was unable to be uploaded to the Arduino, therefore, the connection was changed by connecting the TX on baseboard to the pin D10 of Arduino and the RX on baseboard to the pin D11 of Arduino. Meanwhile, it was defined in the program as shown in Figure 16. Then the code was successfully uploaded to the Arduino.

```
#include<SoftwareSerial.h>
SoftwareSerial mySerial(10,11);//TX,RX
```

Figure 16. Definition Code

Lab Session II:

Task1: Studied about MCU digital I/O, interrupts, Real-Time communication and control.

Concerning knowledge of MCU digital I/O, interrupts, Real-Time communication and control were self-studied.

Task 2: Re-burned the baseboard with a given RT control firmware.

As shown in Figure 17, the latest version of Arduino-1.8.10 was downloaded and installed. The Arduino AVR Boards were updated to version 1.8.1. The ATmega328PB Boards were installed as shown in Figure 18.

Figure 17. Version of Arduino and AVR Boards

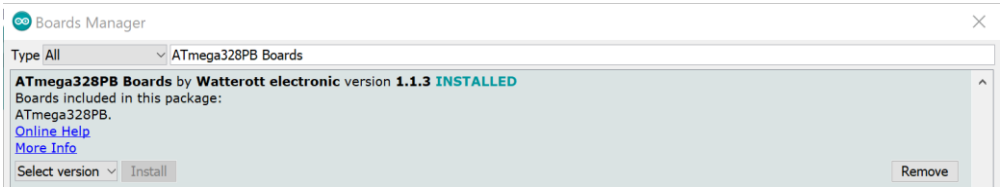


Figure 18. Installation of ATmega328PB Boards

Library support for ATmega328PB was gotten, and related Header files were replaced as shown in Figure 19.

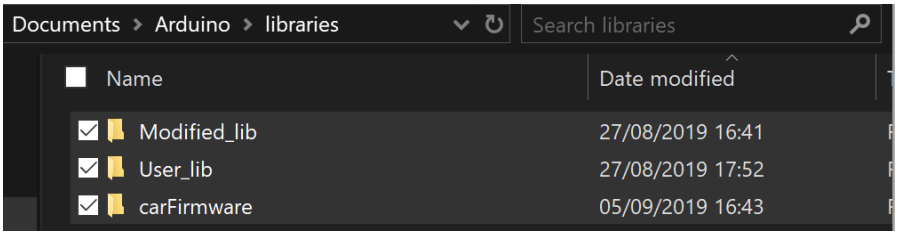


Figure 19. Replacement of concerning Header files

Further reburn actions were not taken due to the limitation of time.

Task 3: Designed and implemented RT communication between the baseboard and the Arduino Nano, and also the RT control. Verified the performance (frequency >= 200Hz) using oscilloscope.

The main code for RT interrupt was shown in Figure 20 which was basically with an interrupt service routine. And the code used for the modification of voltage output on pin for debugging that was connected to the oscilloscope was shown in Figure 21. The test_2 was the pin for debugging.

```
inline void user_init_timer2()
{
  MsTimer2::set((1000 / freq_ctl), timer2isr);
  MsTimer2::start();
}

void timer2isr()//timer2 interrupt service routine (isr)
{
  interrupts();
}
```

Figure 20. Code for RT interrupt

```
void Timer2ISR()
{
  if(judge){digitalWrite(test_2,HIGH);judge=!judge;
  }
  else{digitalWrite(test_2,LOW);judge=!judge;
  }
}
```

Figure 21. Code for square wave

Task 4: Studied logic gates and PWM. Recorded the waveforms of IRS2008 input/output (any channel), and the waveform of the associated motor voltage.

The logic gates and PWM were self-studied.

The circuit used for recording the waveforms of IRS2008 was shown in Figure 22.

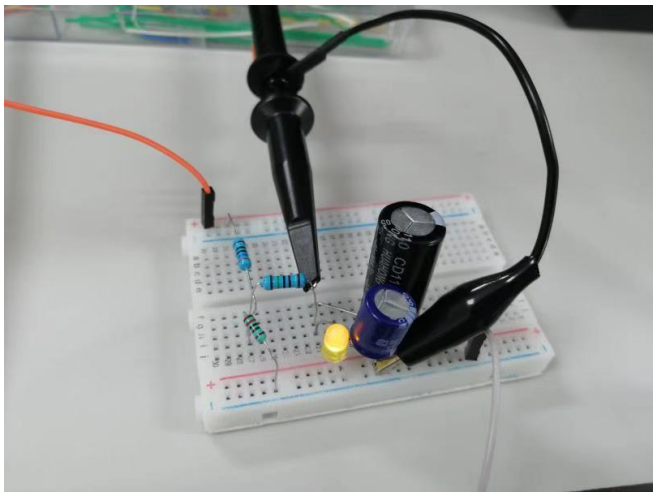


Figure 22. The circuit used for recoding the waveforms of IRS2008

Task5: Studied, implemented and tested different input/output HMI parts with breadboard. Designed the HMI to meet the requirement of demo, soldered and debugged the HMI on a stripboard.

The different input/output HMI parts were studied, implemented and tested with breadboard.

The construction of HMI was followed by the rules in Figure 23.

```
/*
   initialize the library by associating any needed HMI pin
   with the arduino pin number it is connected to
   lcd interface: d2-lcd_d7, d3-lcd_d6, d4-lcd_d5, d5-lcd_d4, d11-lcd_e, d12-lcd_rs
   button switch: d6-bs_1, d7-bs_2
   slide switch: d8-ss
   encoder: d9-enc_1, d10-enc_2, d13-enc_bs
   potentiometer: a0-pm
   microphone: a1-mic
   buzzer: a2-buz
   led_green: a3-led
*/
```

Figure 23. HMI construction rules

Task 6: Integrated the vehicle with the HMI board. Designed the code and debugged for the required challenge. Attempted to complete tasks for the week.

The HMI board was integrated onto the vehicle, and the code was designed and debugged for the required challenge. The tasks for the week were attempted to complete.

The serial communication method used for this session was I2C communication because the UART serial communication method was too slow to process and transmit information and commands for the real time interrupt and the I2C was faster.

Following the schematic of Arduino Nano shown in Figure 24, the connection for I2C serial communication was that VCC, SDA, SCL, GND on baseboard were connected to 5V, A4, A5, GND of Arduino accordingly.

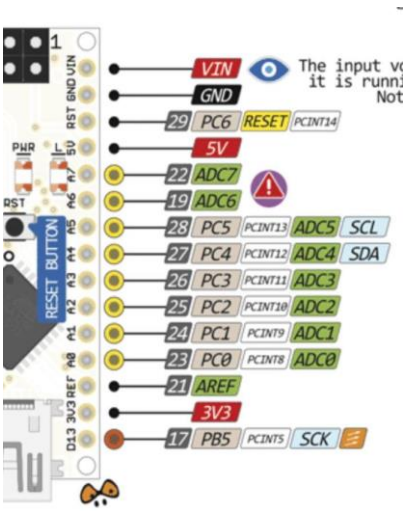


Figure 24. Schematic of Arduino Nano

Results and Discussions

Lab Session I:

Task 1: Potential Divider and H-bridge exercises and tasks.

Potential divider experiments:

In the first attempt, R1 instantly reached the temperature of over 100 ℃ and broke. Because the maximum rated power of R1 was 0.5 W which was lower than the actual power of R1. In the second attempt, the LED did not turn on. Because the current of the whole circuit was too low, therefore, the current flowed through LED was even lower. In the third attempt, the LED successfully turned on when the total voltage on the whole circuit was 12 V, and there was no accidentally broken component. As a result, the circuit was successfully built at the third attempt.

Lesson learnt from the first attempt of breadboard construction was that while using a DC power, the main power switch was turned on firstly and the "Output" power switch remained off. Then, the current imitation was set to prevent the current flowed through components of the circuit reach over the maximum safe rated current and break down the circuit. After the current limitation was set, the "Output" power switch was turned on. Then the voltage was slowly set to fulfil the needs. The DC power flowed out when the green light near the voltage-control switch was on.

H-bridge exercises and tasks:

In the first experiment, the results were recorded in Table 3.

Table 3. Results from the first experiment

	Direction of Rotation	Current of the Motor
1	None	0.01 A
2	Anti-clockwise	0.26 A
3	Clockwise	0.29 A
4	None	0.01 A
5	Anti-clockwise	0.46 A
6	Clockwise	0.57 A

In the second experiment, the results were recorded in Table 4.

Table 4. Results from the second experiment

	Current of the Motor
1	0.01 A
2	0.30 A
3	0.38 A
4	0.01 A
5	0.61 A
6	0.69 A

In the third experiment, the results were recorded in Table 5.

Table 5. Results from the third experiment

Direction of Rotation	Voltage Input	Current of the Motor
Clockwise	1.1 V	0.28 A
Anti-clockwise	1.3 V	0.28 A

Lesson learnt while breadboard constructing was that for a single component, the two feet were not connected onto the same row in the same half part of the breadboard, otherwise, the equipment was shortened therefore broke down the component as well as the circuit.

Results of experiments shown above were some of the results of tasks picked from "H Bridge Project", other tasks were also attempted following the instructions but were not displayed in this report due to the limitation of the length of the report. And there was a failure in this task that the tasks were done too quickly to record the data, therefore, part of the data of the tasks was missing in which case this part of task was incomplete. This was where needed to be improved in future sessions.

Task 2: Got familiar with lab equipment and passed the skill test individually.

The basic operation of lab equipment including multi-meters, oscilloscopes and DC power sources was acquired by each of the group member, and the skill test was passed successfully by each of the group members individually.

Task 3: Got familiar with Arduino and finished the tasks of "Arduino Introduction".

The LED was successfully programmed to blink. And other examples had been attempted but not displayed in this report due to the limitation of the length of the report. After the tasks of "Arduino Introduction" were finished, the basic understanding of the Arduino was acquired by each of the group members.

Task 4: Finished the soldering and debugging of one H-bridge individually. Verified the performance with Arduino software.

Though the 4 H-bridge PCBs seemed soldered fine, but all of them failed the verification. There was no shortened circuit in the PCBs, but the H-bridge was not able to control and drive the motor. Therefore, the possible reason for this problem was that there were several disconnections in the circuit due to which several components were not working properly. And because of the time limitation, the soldered H-bridge PCBs were borrowed for the vehicle. The self-soldered H-bridge PCB was shown in Figure 25.

Figure 25. Self-soldered H-bridge PCB

Task 5: Built the vehicle. Integrated all the components together and attempted to complete tasks for the week.

The integrated vehicle for the week was shown in Figure 26.

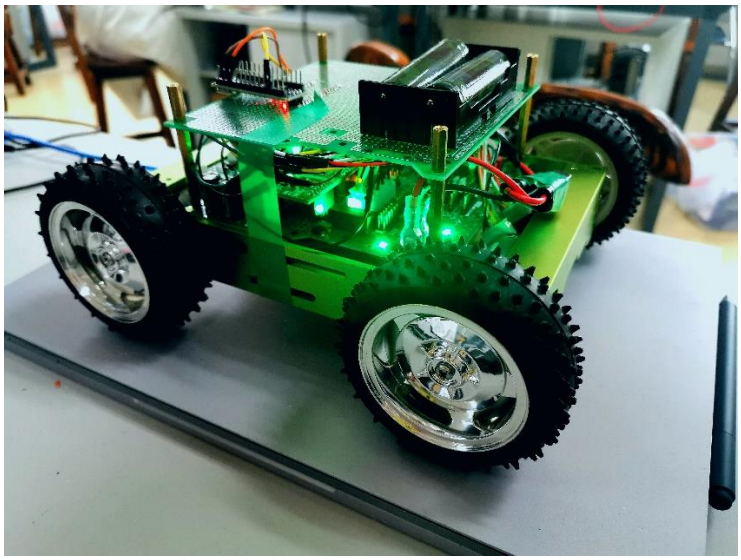


Figure 26. Integrated vehicle for week 1

As a result of the testing part, 4 motors started rotating which marked the baseboard and the motors worked fine.

After the testing, the command "#F020,020,020,020,00468,00468,00468,00468" was given to the vehicle, but the motor 4 was not stopping after rotating as other motors. In further discovery, it was found that the encoder in motor 4 was not working via the command "#Ea". Instead of using a lot time debugging the baseboard and the motor, the algorithm was used to control the motor to complete the tasks avoiding the broken encoder which end up functioning well.

Then the challenging for the final tasks was started. When the vehicle was prepared to be ready for the 10M challenge, several problems occurred. The vehicle was instead of running straight but a little bit to the right. And the mistake was amplified by the 10M long distance. Meanwhile, when the vehicle was tested on the track, it was noticed that the absolute speed of the vehicle was lower than other groups'. To make the vehicle to become one of the fastest, the problems had to be solved due to two simple reasons: the straight line between two spots was always the shortest, and also the faster the absolute speed was the shorter the time cost to run the same distance would be. To solve the first problem, the two motors on the left were set to spin at 97% of the maximum power and the rest were set to spin at 100%. Then when the vehicle was tested, it started to run straight. However, when the old batteries were replaced with new two to solve the second problem, the vehicle was not running straight again. The problem was caused because the larger power the new batteries output caused the difference of speed between wheels of two sides larger than previous so that the vehicle was running to the left. So, the left side wheels were set to spin at 93% of their maximum power, and then the vehicle was running perfectly straight. In the end, the vehicle ran the second fastest which was a great success to gain confidence for the future lab sessions.

In the open lab between 2 lab weeks, the motor was replaced by new one twice and the baseboard was replaced with new one once as well, but the encoder problem still existed. In the further discovery, the possible reason for this problem was that this batch of baseboards had the same unsolvable issue. However, after serial communication method was changed to I2C in the lab session 2, the encoder in motor 4 functioned well.

Lab Session II:

Task1: Studied about MCU digital I/O, interrupts, Real-Time communication and control.

Basic understanding of MCU digital I/O, interrupts, Real-Time communication and control was learnt by each of the group members. However, the effort made to this part of the session was too little to successfully applied the theory to the vehicle resulting in the failure in demo tasks.

Therefore, the improvement for future lab sessions was to pay more attention to rest of the tasks besides the demo tasks.

Task 2: Re-burned the baseboard with a given RT control firmware.

The re-burning task was failed due to the limitation of time. Though the baseboard functioned well, the old version of baseboard firmware was only able to finish few tasks. Therefore, the re-burning of the baseboard was still necessary for future complex tasks. Thus, the re-burning was to be finished before the next lab session.

Task 3: Designed and implemented RT communication between the baseboard and the Arduino Nano, and also the RT control. Verified the performance (frequency >= 200Hz) using oscilloscope.

When the value of the freq_ctl was set to 200, the frequency was what the experiment needed.

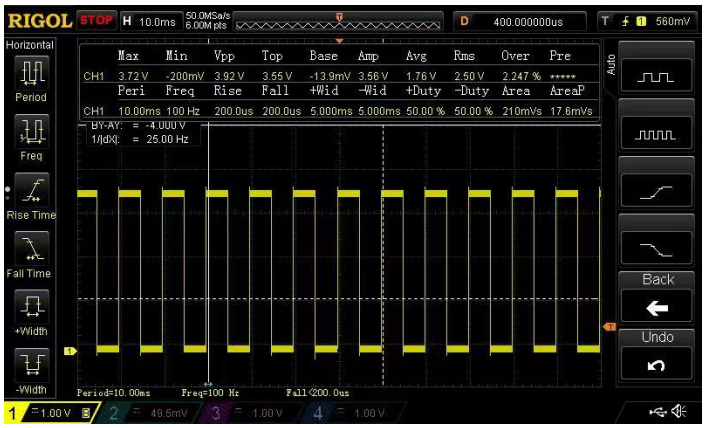


Figure 27. Oscilloscope screenshot for Verification

The frequency shown in the oscilloscope in Figure 27 was the indicate frequency instead of the actual frequency of the interrupt for communication. The actual frequency was twice as the indicate frequency. Therefore, the frequency of 200Hz was the frequency needed for the verification.

Therefore, the program was successfully built with a RT interrupt frequency over 200Hz that was applied to the vehicle in the further experiment.

Task 4: Studied logic gates and PWM. Recorded the waveforms of IRS2008 input/output (any channel), and the waveform of the associated motor voltage.

The basic understanding of logic gates and PWM were learnt by each of the group members.

The waveforms of IRS2008 were recoded as shown in Figure 28 – 31.

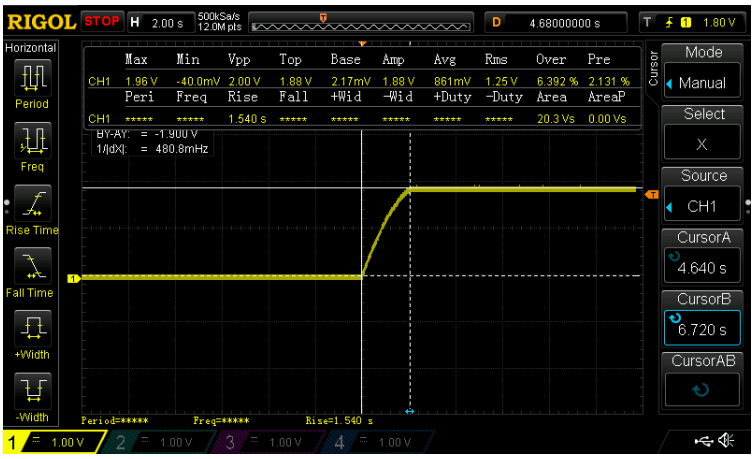


Figure 28. Waveform rise time of IRS2008

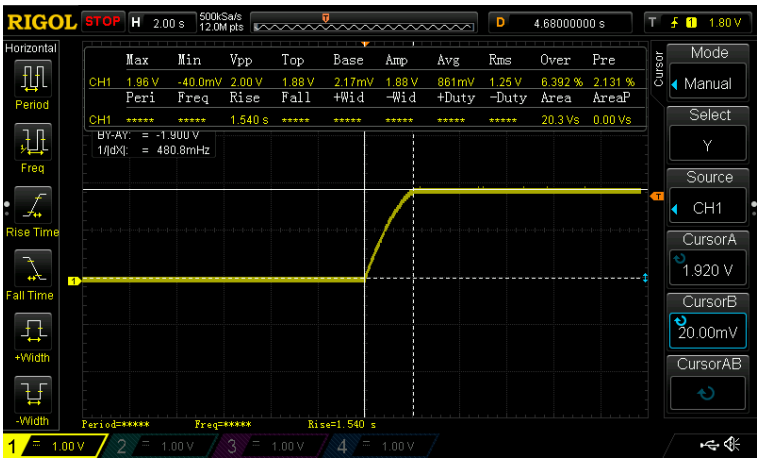


Figure 29. Waveform rise voltage of IRS2008

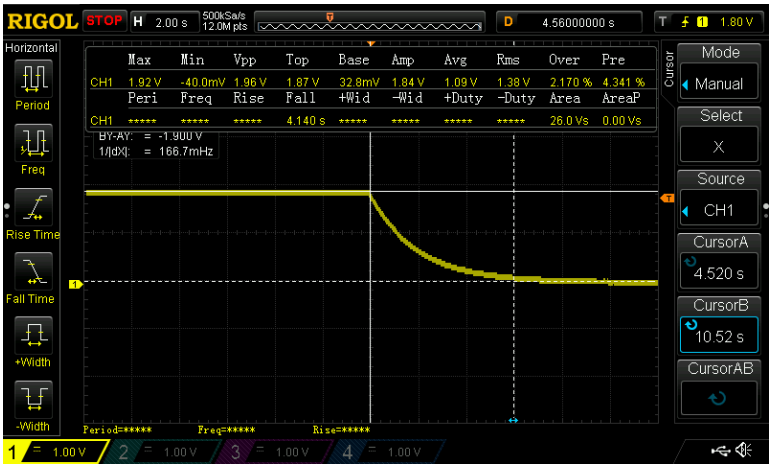


Figure 30. Waveform fall time of IRS2008

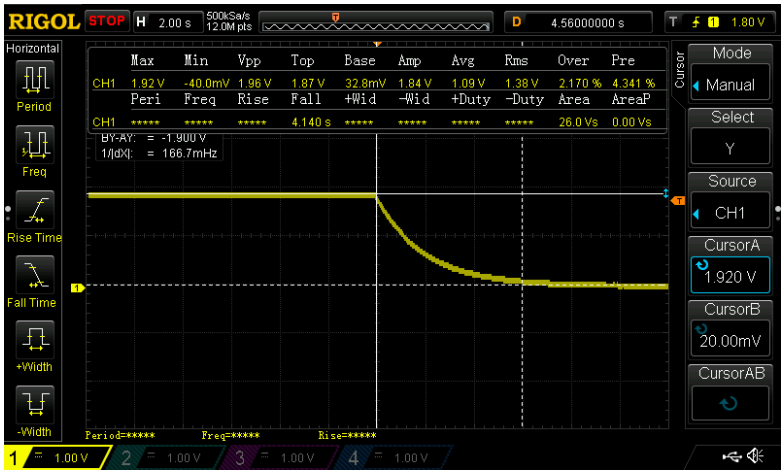


Figure 31. Waveform fall voltage of IRS2008

Task5: Studied, implemented and tested different input/output HMI parts with breadboard. Designed the HMI to meet the requirement of demo, soldered and debugged the HMI on a stripboard.

The integrated HMI board as shown in Figure 32 was designed, soldered and debugged to meet the requirement of demo.

Figure 32. Integrated HMI board

Task 6: Integrated the vehicle with the HMI board. Designed the code and debugged for the required challenge. Attempted to complete tasks for the week.

The demo tasks for the week was failed. One major reason of the failure was the lack of small progresses before a final integration. The construction of circuits and the soldering work for the HMI board were of the best groups. However, even though the HMI was function properly, it was still unable to challenge for the demo in the end. And the reason caused the problems was that the principles to properly function each component of the HMI had not been fully understood before they were integrated together to function for the final demo. Too much attention was paid to the final result while the processes being ignored.

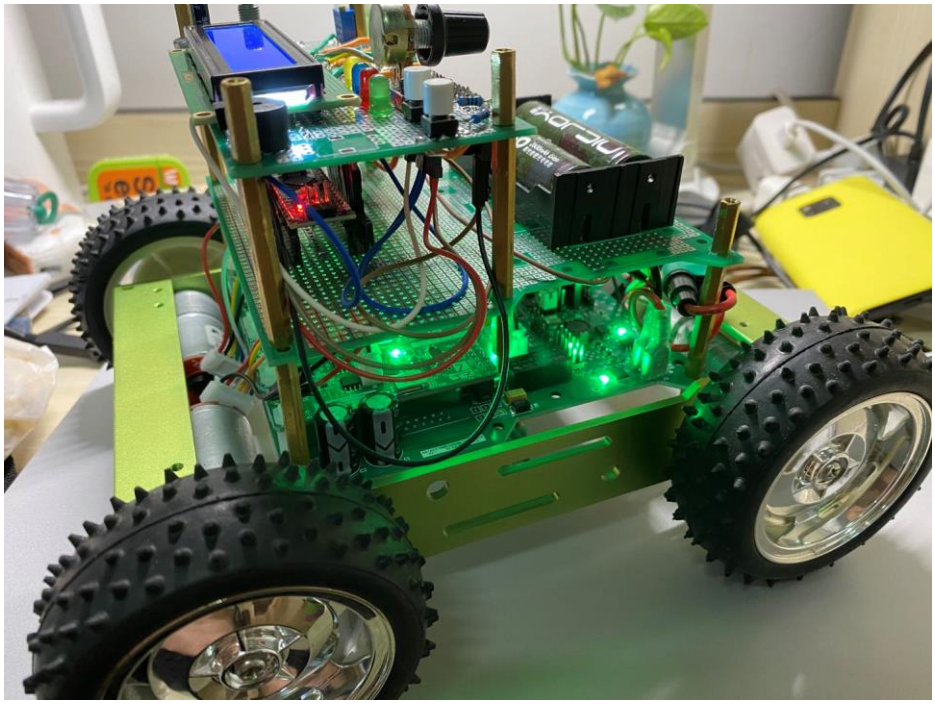


Figure 33. The Final Integrated Vehicle for Session I & II

The final integrated vehicle for session I & II was shown in Figure 33.

Conclusion

During the experiment, session I was about basics of building and testing the vehicle. In the first session, basic understanding of H-bridge and Arduino as well as the basic operation of lab equipment were learnt, and the vehicle was built for simple separate tasks. Session II offered a method to real-time communicate with the vehicle with human machine interface, and an advanced vehicle that was able to finish tasks together was built.

To summarize, these two lab sessions were successful. The demo of session I was a success, the vehicle ran the second fastest in the 10 M race. Though the demo of session II was failed that the vehicle was unable to challenge for the demo tasks successfully in time, the vehicle managed to achieve several tasks and the well-constructed HMI board was designed as well as the verification was passed. Besides demo tasks, other tasks were mostly completed. However, there were things to be improved including time allocation, clarity of tasks and priority. Some record of the data was incomplete which needed improvement.

References

[1] *H61AEE: Project Session I Introduction.* Department of Electrical and Electronic Engineering, University of Nottingham Ningbo China, 2019 [Online].

Available:

https://moodle.nottingham.ac.uk/pluginfile.php/5346356/mod_resource/content/9/Session_1_Introduction%201920%20V1.pdf

[Accessed: 22 November 2019]

[2] *H61AEE: Project Session II Introduction.* Department of Electrical and Electronic Engineering, University of Nottingham Ningbo China, 2019 [Online].

Available:

https://moodle.nottingham.ac.uk/pluginfile.php/5346414/mod_resource/content/16/Session_2_Introduction%201920%20V2.pdf

[Accessed: 22 November 2019]

[3] *H61AEE: Practical electronics: H Bridge Project.* Department of Electrical and Electronic Engineering, University of Nottingham Ningbo China, 2019 [Online].

Available:

https://moodle.nottingham.ac.uk/pluginfile.php/5346367/mod_resource/content/4/H%20Bridge%20Project%201920%20V1.pdf

[Accessed: 22 November 2019]

[4] *H61AEE: Arduino Introduction.* Department of Electrical and Electronic Engineering, University of Nottingham Ningbo China, 2019 [Online].

Available:

https://moodle.nottingham.ac.uk/pluginfile.php/5346380/mod_resource/content/4/01%20Arduino_Introduction%201920.pdf

[Accessed: 22 November 2019]

Appendices

Code for spinning task of session I:

```
#include<SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); //TX,RX

void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  Serial.begin(57600);
  mySerial.begin(57600);
  digitalWrite(13, HIGH);
  delay(3000);
}

void loop() {
  // put your main code here, to run repeatedly:
  mySerial.write("#Baffrr040,040,040,040");
  delay(1660);
  mySerial.write("#ha");
  digitalWrite(13, HIGH);
  delay(1000000);
}
```

Code for square task of session I:

```
#include<SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); //TX,RX

int count = 0;

void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  Serial.begin(57600);
  mySerial.begin(57600);
  digitalWrite(13, HIGH);
  delay(3000);
}

void loop() {
  // put your main code here, to run repeatedly:
  mySerial.write("#Baffff040,040,040,040");
  delay(2600);
  mySerial.write("#ha");
  delay(1000);
  mySerial.write("#Baffrr040,040,040,040");
  delay(400);
  count++;
  if (count == 4) {
    mySerial.write("#ha");
    digitalWrite(13, HIGH);
    delay(1000000);
  }
}
```

Code for 10 M race task of session I:

```
#include<SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); //TX,RX

void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  Serial.begin(57600);
  mySerial.begin(57600);
  digitalWrite(13, HIGH);
  delay(5000);
}

void loop() {
  // put your main code here, to run repeatedly:
  mySerial.write("#Baffff093,093,099,099");
  delay(7600);
  mySerial.write("#ha");
  digitalWrite(13, HIGH);
  delay(1000000);
}
```

Code for verification of session II:

```
#include<Wire.h>
#include<MsTimer2.h>

#include<LiquidCrystal.h>//include the library code

#define TEN_METER (11129)//234.256*10/0.067*3.1415926)
long unsigned int encoder1Value = 0;
long unsigned int encoder2Value = 0;
long unsigned int encoder3Value = 0;
long unsigned int encoder4Value = 0;

long unsigned int target1 = 0;

#define debug_IO (12)

long unsigned int direction_vehicle = 1;
long unsigned int flag_direction = 1;

long unsigned int flag_debugIO = 0;
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // put your setup code here, to run once:
  long unsigned int i = 0;

  pinMode(13, OUTPUT); //Configure the reset pin as an output and hold the robot in reset
  digitalWrite(13, LOW);

  noInterrupts();
```

```
 //Setup the serial ports
 Serial.begin(57600);//This is the hardware port connected to the PC through the USB connection
 Wire.begin();
 target1 = 0x80000000 + TEN_METER;

 User_init_timer2();
 pinMode(debug_IO, OUTPUT);
 interrupts();//enable

 //wait 5 seconds
 delay(5000);

 //Serial.println("software serial simple test!");//Send some text to the PC
 //set up the LCD's number of columns and rows:
 lcd.begin(16, 2);
 //Print a message to the LCD
 //lcd.print("MARK_II");
 digitalWrite(13, HIGH); //Release the robot from reset
 delay(100);//A short delay to allow the robot to start-up
 Wire.beginTransmission(42);
 Wire.write("sa");
 for (i = 1; i <= 4; i++) {
   Wire.write(50);
   Wire.write(0);
 }
 Wire.endTransmission();

 Wire.beginTransmission(42);
 Wire.write("daffff");
 Wire.endTransmission();
}
```

```
void loop() {
  // put your main code here, to run repeatedly:

  /*delay(10);
    readEncoder();*/
  /*Serial.println("Encoder 1 read text!");//Send some text to the PC
    Serial.println(encoder1Value);//Send some text to the PC
    Serial.println("Encoder 2 read text!");//Send some text to the PC
    Serial.println(encoder2Value);//Send some text to the PC
    Serial.println("Encoder 3 read text!");//Send some text to the PC
    Serial.println(encoder3Value);//Send some text to the PC
    Serial.println("Encoder 4 read text!");//Send some text to the PC
    Serial.println(encoder4Value);//Send some text to the PC
    delay(500);

    if (encoder1Value > 0x80040000)
    {
    encoder1Value = 0x80000000;
    }
    else if (encoder1Value < 0x7FFC0000)
    {
    encoder1Value = 0x80000000;
    }

    if (encoder1Value > target1)
    {
    //Stop all the motors
    Wire.beginTransmission(42);
    Wire.write("ha");
    Wire.endTransmission();
    digitalWrite(13, HIGH);
```

```
  while (1);
  }*/
 /*if (direction_vehicle)
  {
  Wire.beginTransmission(42);
  Wire.write("daffff");
  Wire.endTransmission();
  }
  else if (!direction_vehicle)
  {
  Wire.beginTransmission(42);
  Wire.write("darrrr");
  Wire.endTransmission();
  }*/
}

void readEncoder()
{
 long unsigned int encoder1 = 0;
 long unsigned int encoder2 = 0;
 long unsigned int encoder3 = 0;
 long unsigned int encoder4 = 0;
 Wire.beginTransmission(42);
 Wire.write("i");
 Wire.endTransmission();
 //delay(1);
 Wire.requestFrom(42, 8);
 //delay(10);
 //if(Wire.avaliable()=8)
 {
  encoder1 = (long unsigned int)Wire.read();
  encoder1 += ((long unsigned int)Wire.read() << 8);
```

```
    encoder1 += ((long unsigned int)Wire.read() << 16);
    encoder1 += ((long unsigned int)Wire.read() << 24);
    encoder2 = (long unsigned int)Wire.read();
    encoder2 += ((long unsigned int)Wire.read() << 8);
    encoder2 += ((long unsigned int)Wire.read() << 16);
    encoder2 += ((long unsigned int)Wire.read() << 24);
  }
  encoder1Value = encoder1;
  encoder2Value = encoder2;
  Wire.requestFrom(42, 8);
  //delay(10);
  //if(Wire.avaliable()=8)
  {
    encoder3 = (long unsigned int)Wire.read();
    encoder3 += ((long unsigned int)Wire.read() << 8);
    encoder3 += ((long unsigned int)Wire.read() << 16);
    encoder3 += ((long unsigned int)Wire.read() << 24);
    encoder4 = (long unsigned int)Wire.read();
    encoder4 += ((long unsigned int)Wire.read() << 8);
    encoder4 += ((long unsigned int)Wire.read() << 16);
    encoder4 += ((long unsigned int)Wire.read() << 24);
  }
  encoder3Value = encoder3;
  encoder4Value = encoder4;
}

inline void User_init_timer2()
{
  MsTimer2::set(5, Timer2ISR);
  MsTimer2::start();
}
```

```
void Timer2ISR()//Timer2 interrupt service routine (ISR)
{
  interrupts();
  if (!flag_debugIO) {
    flag_debugIO = !flag_debugIO;
    digitalWrite(debug_IO, HIGH);
  }
  else {
    digitalWrite(debug_IO, LOW);
    flag_debugIO = !flag_debugIO;
  }

  PC_print();
}

inline void PC_print()
{
  long static unsigned int flag_500ms = 0;
  long static unsigned int counter_500ms = 0;

  //delay(10);
  readEncoder();

  if (counter_500ms < 200)counter_500ms++;
  else
  {
    counter_500ms = 0;
    Serial.println("Print every 1.0s\n");
    direction_vehicle = !direction_vehicle;
    Serial.println(encoder1Value);
  }
```

```
  if (direction_vehicle && !flag_direction)
  {
    Wire.beginTransmission(42);
    Wire.write("daffff");
    Wire.endTransmission();
    flag_direction = !flag_direction;
  }
  else if (!direction_vehicle && flag_direction)
  {
    Wire.beginTransmission(42);
    Wire.write("darrrr");
    Wire.endTransmission();
    flag_direction = !flag_direction;
  }
}
```

Code for demo of session II: (Not debugged)

```
/*
  initialize the library by associating any needed HMI pin
  with the arduino pin number it is connected to
  lcd interface: d2-lcd_d7, d3-lcd_d6, d4-lcd_d5, d5-lcd_d4, d11-lcd_e, d12-lcd_rs
  button switch: d6-bs_1, d7-bs_2
  slide switch: d8-ss
  encoder: d9-enc_1, d10-enc_2, d13-enc_bs
  potentiometer: a0-pm
  microphone: a1-mic
  buzzer: a2-buz
  led_green: a3-led
*/

#include<Wire.h>
#include<LiquidCrystal.h>
#include<MsTimer2.h>
#include<PinChangeInterrupt.h>

#define lcd_d7 (2)
#define lcd_d6 (3)
#define lcd_d5 (4)
#define lcd_d4 (5)
#define lcd_e (11)
#define lcd_rs (12)
LiquidCrystal lcd(lcd_rs, lcd_e, lcd_d4, lcd_d5, lcd_d6, lcd_d7);

#define bs_1 (6)
#define bs_2 (7)
#define ss (8)
#define enc_1 (9)
#define enc_2 (10)
#define enc_bs (13)
```

```
#define pm (A0)
#define mic (A1)
#define buz (A2)
#define led (A3)

#define freq_ctl (200)

unsigned char flag_direction = 0;
unsigned char flag_task = 1;
unsigned char flag_status = 0;
unsigned char flag_lcd = 0;
unsigned char flag_mode = 0;
unsigned char flag_lcd_mode = 0;
unsigned char flag_led = 0;
unsigned char flag_led_mode = 0;
unsigned char flag_led_blink = 0;

unsigned int vehicle_speed = 20;
unsigned int vehicle_target = 1000;
unsigned int vehicle_progress = 0;
unsigned int speed_min = 20;
unsigned int speed_max = 60;
unsigned int counter_led = 0;

long unsigned int encoder1Value = 0;
long unsigned int encoder2Value = 0;
long unsigned int encoder3Value = 0;
long unsigned int encoder4Value = 0;

void setup() {
  // put your setup code here, to run once:

  pinMode(13, OUTPUT); //configure the reset pin as an output and hold the robot in reset
  digitalWrite(13, LOW);
```

```
noInterrupts();

//setup the serial ports
Serial.begin(57600);//this is the hardware port connected to the PC through the USB connection
Wire.begin();

//configure pins
pinMode(bs_1, INPUT_PULLUP);
pinMode(bs_2, INPUT_PULLUP);
pinMode(ss, INPUT_PULLUP);
pinMode(enc_1, INPUT);
pinMode(enc_2, INPUT);
//pinMode(enc_bs, INPUT_PULLUP);
digitalWrite(enc_1, HIGH);
digitalWrite(enc_2, HIGH);
pinMode(buz, OUTPUT);
pinMode(led, OUTPUT);

digitalWrite(13, HIGH); //release the robot from reset
delay(100);//a short delay to allow the robot to start-up

lcd.begin(16, 2); //set up the lcd's number of columns and rows
lcd.print("MARK_1");//print a message to the lcd

delay(1000);

user_init_timer2();
interrupts();//enable

attachPinChangeInterrupt(digitalPinToPCINT(ss), switch_direction, CHANGE);
attachPinChangeInterrupt(digitalPinToPCINT(bs_1), switch_task, RISING);
attachPinChangeInterrupt(digitalPinToPCINT(bs_2), switch_status, RISING);
//attachInterrupt(interrupt0, increase, RISING);
```

```
  //attachPinChangeInterrupt(digitalPinToPCINT(enc_2), decrease, RISING);
  //attachPinChangeInterrupt(digitalPinToPCINT(enc_bs), switch_mode, RISING);
}

void loop() {
  // put your main code here, to run repeatedly:
  //for test only
  //Serial.println(speed_max);
}

inline void user_init_timer2()
{
  MsTimer2::set((1000 / freq_ctl), timer2isr);
  MsTimer2::start();
}

void timer2isr()//timer2 interrupt service routine (isr)
{
  interrupts();

  /*static unsigned char flag_led = 0;
    static unsigned int counter_1s = 0;*/

  /*if (digitalRead(enc_1) == LOW)
    {
    if (digitalRead(enc_2) == LOW)increase();
    }
    else
    {
    if (digitalRead(enc_2) == LOW)decrease();
    }*/

  vehicle();
```

```
 /*if(counter_1s<(freq_ctl>>1))//led blinking to make sure the baseboard is working well
   {
   counter_1s++;
   }
   else
   {
   counter_1s=0;
   if(!flag_led)
   {
     flag_led=!flag_led;
     digitalWrite(13,HIGH);//turn the led on (HIGH is the voltage level)
   }
   else
   {
     flag_led=!flag_led;
     digitalWrite(13,LOW);//turn the led off by making the voltage LOW
   }
   }*/
}

inline void vehicle()
{
 /*static unsigned char flag_500ms=0;
   static unsigned int counter_500ms=0;

   if(counter_500ms<(freq_ctl>>2))
   {
   counter_500ms++;
   }
   else
   {
   counter_500ms=0;
   if(!flag_500ms)
   {
```

```
    flag_500ms=!flag_500ms;
    Serial.println("Print every 0.5s\n");
  }
  else flag_500ms=!flag_500ms;
  }*/

//setup led
if (!flag_status)digitalWrite(led, LOW);
else if (flag_led && !flag_led_mode)led_solidON();
else if (flag_led && flag_led_mode)led_blink(500);

/*//set speed limit via reading potentiometer value
  set_speedlimit();

  //read encoder value
  readEncoder();

  if (encoder1Value > vehicle_target)flag_status = 0;*/

//vehicle_status control
if (!flag_status)
{
  Wire.beginTransmission(42);
  Wire.write("ha");
  Wire.endTransmission();
  //return;
}

//setup the speed, direction for vehicle
if (flag_status)
{
  Wire.beginTransmission(42);
  Wire.write("sa");
  for (int i = 1; i <= 4; i++)
```

```
  {
    Wire.write(vehicle_speed);
    Wire.write(0);
  }
  Wire.endTransmission();
}
switch (flag_task)
{
  case (0): //task1: spin
    {
      flag_led_mode = 0;
      switch (flag_direction)
      {
        case (0): //clockwise
          {
            if (flag_lcd && !flag_lcd_mode)
            {
              lcd.clear();
              lcd.setCursor(0, 0); //set the cursor to column 0, line 0
              lcd.print("Task_1");//printf the number of tasks
              lcd.setCursor(0, 1); //set the cursor to column 0, line 1
              lcd.print("Clockwise");//printf the direction
              flag_lcd = 0;
            }
            if (!flag_status)break;
            Wire.beginTransmission(42);
            Wire.write("daffrr");
            Wire.endTransmission();
            break;
          }
        case (1): //anti-clockwise
          {
            if (flag_lcd && !flag_lcd_mode)
            {
```

```
            lcd.clear();
            lcd.setCursor(0, 0); //set the cursor to column 0, line 0
            lcd.print("Task_1");//printf the number of tasks
            lcd.setCursor(0, 1); //set the cursor to column 0, line 1
            lcd.print("Anti-Clockwise");//printf the direction
            flag_lcd = 0;
          }
          if (!flag_status)break;
          Wire.beginTransmission(42);
          Wire.write("darrff");
          Wire.endTransmission();
          break;
        }
    }
    break;
  }
case (1): //task3: run straight
  {
    flag_led_mode = 1;
    switch (flag_direction)
    {
      case (0): //forward
        {
          if (flag_lcd && !flag_lcd_mode)
          {
            lcd.clear();
            lcd.setCursor(0, 0); //set the cursor to column 0, line 0
            lcd.print("Task_2");//printf the number of tasks
            lcd.setCursor(0, 1); //set the cursor to column 0, line 1
            lcd.print("Forward");//printf the direction
            flag_lcd = 0;
          }
          if (!flag_status)break;
          Wire.beginTransmission(42);
```

```
          Wire.write("daffff");
          Wire.endTransmission();
          break;
         }
       case (1): //backward
         {
          if (flag_lcd && !flag_lcd_mode)
          {
           lcd.clear();
           lcd.setCursor(0, 0); //set the cursor to column 0, line 0
           lcd.print("Task_2");//printf the number of tasks
           lcd.setCursor(0, 1); //set the cursor to column 0, line 1
           lcd.print("Backward");//printf the direction
           flag_lcd = 0;
          }
          if (!flag_status)break;
          Wire.beginTransmission(42);
          Wire.write("darrrr");
          Wire.endTransmission();
          break;
         }
     }
     break;
   }
}

//lcd for speed & target setup
switch (flag_mode)
{
  case (0): //speed setup
    {
     if (flag_lcd && flag_lcd_mode)
     {
       lcd.clear();
```

```
        lcd.setCursor(0, 0);
        lcd.print("Speed:");
        lcd.setCursor(0, 1);
        lcd.print(vehicle_speed);
      }
      break;
    }
  case (1): //target setup
    {
      if (flag_lcd && flag_lcd_mode)
      {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Target:");
        lcd.setCursor(0, 1);
        lcd.print(vehicle_target);
      }
      break;
    }
}

//lcd display speed and progress while vehicle running
if (flag_status && flag_lcd)
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Speed:");
  lcd.setCursor(8, 0);
  lcd.print(vehicle_speed);
  lcd.setCursor(0, 1);
  lcd.print("Progress:    %");
}
vehicle_progress = (int)encoder1Value * 100 / vehicle_target; //calculate progress
if (flag_status)
```

```
  {
    lcd.setCursor(11, 1);
    lcd.print(vehicle_progress);
  }
}

void switch_direction()
{
  flag_direction = !flag_direction;
  flag_lcd = 1;
  flag_status = 0;
}

void switch_task()
{
  flag_task = !flag_task;
  flag_lcd = 1;
  flag_status = 0;
  flag_led = 1;
}

void switch_status()
{
  flag_status = !flag_status;
  flag_lcd = 1;
}

void switch_mode()
{
  flag_mode = !flag_mode;
  flag_lcd = 1;
  flag_status = 0;
}
```

```
void increase()
{
  flag_status = 0;
  flag_lcd_mode = 1;
  flag_lcd = 1;
  switch (flag_mode)
  {
    case (0): //speed setup
      {
        vehicle_speed++;
        if (vehicle_speed > speed_max)vehicle_speed = speed_max;
        break;
      }
    case (1): //target setup
      {
        vehicle_target++;
        break;
      }
  }
}

void decrease()
{
  flag_status = 0;
  flag_lcd_mode = 1;
  flag_lcd = 1;
  switch (flag_mode)
  {
    case (0): //speed setup
      {
        vehicle_speed--;
        if (vehicle_speed < speed_min)vehicle_speed = speed_min;
        break;
      }
```

```
    case (1): //target setup
      {
        vehicle_target--;
        if (vehicle_target < 0)vehicle_target = 0;
        break;
      }
  }
}

void readEncoder()
{
  long unsigned int encoder1 = 0;
  long unsigned int encoder2 = 0;
  long unsigned int encoder3 = 0;
  long unsigned int encoder4 = 0;

  Wire.beginTransmission(42);
  Wire.write("i");
  Wire.endTransmission();

  Wire.requestFrom(42, 8);
  encoder1 = (long unsigned int)Wire.read();
  encoder1 += (long unsigned int)Wire.read() << 8;
  encoder1 += (long unsigned int)Wire.read() << 16;
  encoder1 += (long unsigned int)Wire.read() << 24;
  encoder2 = (long unsigned int)Wire.read();
  encoder2 += (long unsigned int)Wire.read() << 8;
  encoder2 += (long unsigned int)Wire.read() << 16;
  encoder2 += (long unsigned int)Wire.read() << 24;
  encoder1Value = encoder1;
  encoder2Value = encoder2;
  Wire.requestFrom(42, 8);
  encoder3 = (long unsigned int)Wire.read();
  encoder3 += (long unsigned int)Wire.read() << 8;
```

```
  encoder3 += (long unsigned int)Wire.read() << 16;
  encoder3 += (long unsigned int)Wire.read() << 24;
  encoder4 = (long unsigned int)Wire.read();
  encoder4 += (long unsigned int)Wire.read() << 8;
  encoder4 += (long unsigned int)Wire.read() << 16;
  encoder4 += (long unsigned int)Wire.read() << 24;
  encoder3Value = encoder3;
  encoder4Value = encoder4;

  if (encoder1Value > 0x80040000)encoder1Value = 0x80000000;
  else if (encoder1Value < 0x7FFC0000)encoder1Value = 0x80000000;
}

void set_speedlimit()
{
  long unsigned int pmValue = 0;

  pmValue = analogRead(pm);
  Serial.println(pmValue);//for test only
  speed_max = (int)(pmValue + 1) * 20 / 1024 + 60;
}

void led_solidON()
{
  flag_led = 0;
  digitalWrite(led, HIGH);
}

void led_blink(int led_freq)
{
  //flag_led = 0;

  if (counter_led < led_freq * freq_ctl / 2000)
  {
```

```
    counter_led++;
  }
  else
  {
    counter_led = 0;
    if (!flag_led_blink)
    {
      flag_led_blink = !flag_led_blink;
      digitalWrite(led, HIGH); //turn the led on (HIGH is the voltage level)
    }
    else
    {
      flag_led_blink = !flag_led_blink;
      digitalWrite(led, LOW); //turn the led off by making the voltage LOW
    }
  }
}
```