**(|CSEN701|) Embedded System Architecture Project (APA FI RS)**

**Team: (64 Stardust)**

Alaa Hesham Mohamed ,49-20123, T23

Doaa Ahmed ,49-18923, T16

Andrew Ossama Thabet Henien Sky ,49-2449, T12

Omar Muhammad Alsaied Dawood ,49-11483, T12

Youssef Samy Andraous Ghattas ,40-20092, T12

# 1.Project Description

Our aim from this project is to make a car that we can control using Bluetooth module and Dabble app. Dabble app contains 4 buttons to move the car in four directions: forward, backward, right, and left. It also has a start button, once we press this button the car starts performing parallel parking autonomously. For obstacle detection, a buzzer is used to alert us whenever ultrasonic sensors detect close obstacles.

The car also has a radio system that can be controlled by a touch screen, a fuel indicator to know the level of fuel in the car, and rain wipers that work when the rain sensor indicates that there is rain.

# 2.Components

1. Bluetooth module to connect the car with dabble app to control it.
2. Rain sensor to tell the wipers that there is rain and they must work.
3. Servo motor ,representing the wipers, that moves back and forth whenever the rain sensor indicates that there is rain.
4. Water sensor to know the level of fuel.
5. 7-segment display to display fuel level 0->empty 1->low 2->high.
6. Four dc motors to move the car’s 4 wheels.
7. Motor driver to control the 4 dc motors of the 4 wheels.
8. Lcd touch screen to control the radio system.
9. Radio module to play radio channels.
10. Arduino uno board.
11. Arduino mega board.
12. Three ultra-sonic sensors to calculate the distance to nearby objects in order to perform parking and alerting for obstacles.

# 

# 3. Libraries

1. Name: GAMEPAD\_MODULE, Dabble

*Functions:*

*1.Dabble.begin(9600); // specifying baud rate*

*2.Dabble.processInput(); //To refresh the data that the Arduino UNO got from the app*

*3.GamePad.isUpPressed(); // Checks if UP is pressed in the gamepad.*

*4.GamePad.isDownPressed(); // Checks if DOWN is pressed in the gamepad*

*5.GamePad.isLeftPressed(); // Checks if LEFT is pressed in the gamepad.*

*6.GamePad.isRightPressed(); // Checks if RIGHT is pressed in the gamepad. 7. GamePad.isStartPressed(); // Checks if start is pressed*

2. Name: Servo

*Functions:*

1. *tap\_servo. attach(tap\_servo\_pin);// specifying input pin* 2. *tap\_servo.write(degree); //specify servo degree*

3.Name: Adafruit\_GFX\_Button

*Functions:*

1. *Adafruit\_GFX\_Button on\_btn // initiate button*
2. *off\_btn.press/// sets button pressed*

*3.btn.contains(pixel\_x, pixel\_y); // check if btn contains the points x,y 4. off\_btn.justReleased() //sets button released*

4. Name: MCUFRIEND\_kbv, TouchScreen

Functions:

1. *MCUFRIEND\_kbv tft; //initiate screen*
2. *tft.setRotation(0); //PORTRAIT*
3. *tft.fillScreen(BLACK);*
4. *tft.setCursor(60, 50);//specify where we will write*
5. *tft.fillRect(55, 45, 60, 20, BLACK); // draw rectangle* 6 *tft.print("OFF"); //print on screen*

5. Name: TEA5767

Functions:

1. *radio.setFrequency(frequency);//start the radio with this channel*
2. *radio.setMuted(true/false); //mute-unmute radio receiver*
3. *TEA5767 radio = TEA5767(); // initiate radio*

# 5. Inputs

1. From water sensor analogRead(respin); //Read data from analog pin respin=A15
2. From rain sensor: int sensor\_pin=31; pinMode(sensor\_pin, INPUT);
3. From touch screen.
4. From Bluetooth module.

How did we handle each input

1. For the rain sensor, when it detects any water, the wipers function is then executed.

2. For the fuel sensor it takes the levels of water from 0 -800 so we divided 3 different if statements to detect if it’s low (0-200) medium(201-600) high(60-800)

3. For the screen, it detects where exactly the user touched it and performs the proper function based on whether the user touched the On/ off button, the next button or the previous button.

4. The remote directions from the user using the Bluetooth module.

# 6. Outputs

To servo motor : int tap\_servo\_pin = 30; tap\_servo.attach(tap\_servo\_pin); tap\_servo.write(0); tap\_servo.write(180); To 7-segment display:

int a = 49; int b = 47; int c = 43; int d = 39; int e = 41; int f = 45; int g = 51;

pinMode(a, OUTPUT); pinMode(b, OUTPUT); pinMode(c, OUTPUT); pinMode(d, OUTPUT); pinMode(e, OUTPUT); pinMode(f, OUTPUT); pinMode(g, OUTPUT); void

turn\_off\_all\_segs() { int seg\_len =

7; int segs[seg\_len] = { a, b, c, d, e, f, g }; for (byte i =

0; i < seg\_len; i++) { digitalWrite(segs[i], LOW); // turn off the seg

} } void display\_dig(int \*arr, int arr\_len) {

turn\_off\_all\_segs(); for (byte i = 0; i < arr\_len; i++) { digitalWrite(arr[i], HIGH);

}

}

void display\_num(int num) {

// given a num, display the num on the seven segment display

// define the segments that map to each pin int one\_dig[2] = { b, c }; int two\_dig[5]

= { a, b, g, e, d }; int zero\_dig[6] = { a, b, c, d, e, f };

turn\_off\_all\_segs();

switch (num) { case 0: display\_dig(zero\_dig, 6); break; case 1:

display\_dig(one\_dig, 2); break; case 2: display\_dig(two\_dig, 5); break;

}

}

# 7. How we handled the output

First for the rain sensor, we get the input which is that there is rain so we can call the function that moves the servo motor 180 degrees. then we have the fuel which takes the input either low medium, or high, and print it in the 7 segments as a scale from 0-2 (0 is low, 2 is high)

# 8. Limitations

We had some trouble running FreeRTOS, the project was working really good before dividing the system into tasks with priorities.

Using the Mega board to handle the screen was one of the challenges we faced. As the screen only displayed but didn’t receive the touching. We realized then that the screen needed calibration and used the proper Example to do so.

# 9. Our work

We all worked together most of the time.