IOT RC CAR

The RC car system is visualized by the following diagram:

Afbeelding met tekst, diagram, schermopname, Plan

Automatisch gegenereerde beschrijving

Figure 1: Schematic of the RC CAR

The project has two subsystems. The controller subsystem and the car subsystem. Both utilize a PSOC. The subsystems communicate using Bluetooth low energy (BLE).

# Controller subsystem

The controller subsystem utilizes a PSoC 6 CY8CKIT-062-WIFI-BT. This subsystem controls the movement of the car. The input for this is a single joystick that is read out by the PSoC using the on board 12-bit ADC. The subsystem also two seven-segment displays to show the speed of the car in km/h. The joystick readout is sent via BLE to the car subsystem. The BLE communication is bidirectional, so the controller also receives the speed of the car to display. The PSoC uses RTOS to schedule the different tasks.

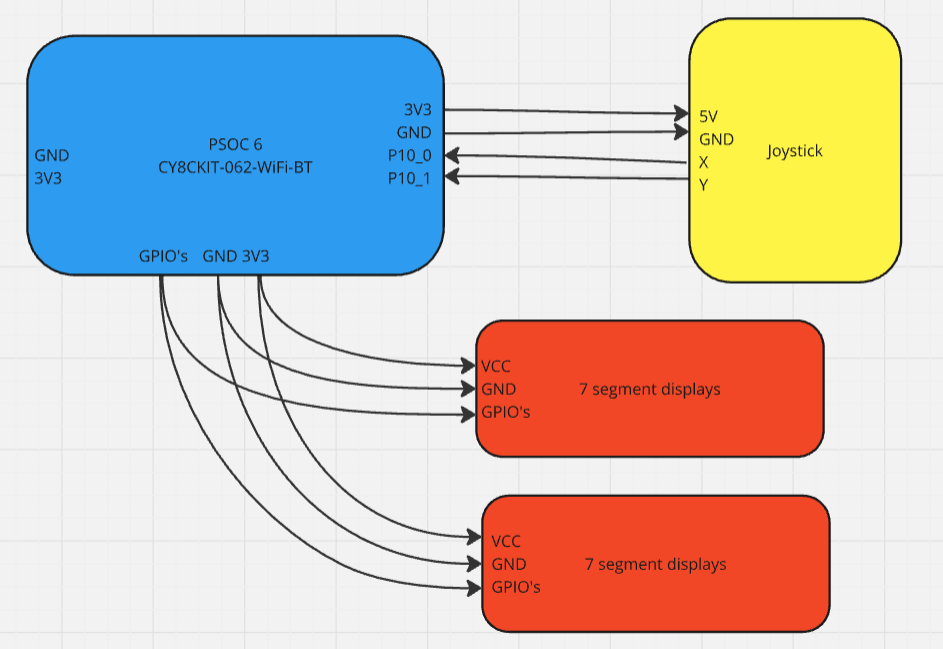


Figure 2: Controller subsystem wiring diagram

# Car subsystem

The car subsystem uses the CY8PROTO-062-4343W PSoC 6. This PSoC is connected to the controller via BLE. It is connected to a MPU6050 breakout board which has an accelerometer, a gyroscope and a temperature sensor. This interfacing is done via I2C. The PSoC reads this data and pushes it to the user interface via MQTT over the HiveMQ broker. This is a secure connection. This system also uses RTOS for scheduling.

The PSoC drives the H-bridge breakout board. This board features a L293D H-bridge IC for controlling two motors. This is ideal because the car frame used in the project had two drivable DC motors.

The car subsystem is powered by a 12V rechargeable Lithium battery. The voltage was lowered to 5V using a DC/DC step down converter for powering the PSoC and H-bridge IC. However, during the construction of the process the 12V to 5V DC/DC converter broke and instead the 5V voltage regulator on the motor driver breakout board is used to power the PSoC. The motors themselves are driven with 12V from the battery.

All of this is soldered onto a PCB to have a cleaner design than just a breadboard. This board is depicted in figure 3.

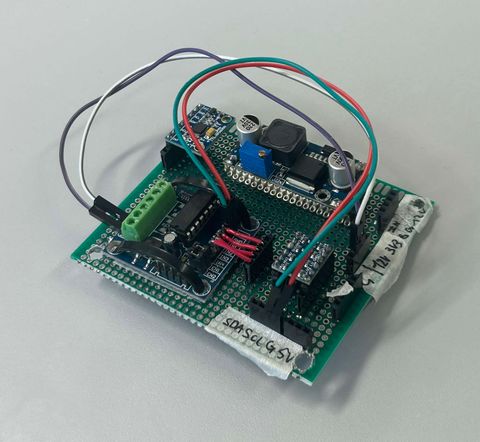


Figure 3: Car subsystem circuit board

Figure 4 shows the wiring diagram of the car subsystem.

Afbeelding met tekst, diagram, Plan, schermopname

Automatisch gegenereerde beschrijving

Figure 4: Wiring diagram car subsystem

# Server UI

The web user interface runs on a backend python flask application. The user interface displays the MPU6050 data and has a cube that rotates and simulates the orientation of the car. The speed in m/s and km/h are also displayed along with the rpm. The UI has three commands that it can send to the car PSoC. These are one button to toggle the motors. With this the motors can be disabled via secure MQTT connection. The second is a reset of the car’s orientation. If the orientation on the UI does not match the car’s orientation, then with this reset button they can be matched again. This can happen due to slow data communication with MQTT. And the last is to enter low power mode.

Afbeelding met tekst, schermopname, nummer, Lettertype

Automatisch gegenereerde beschrijving

Figure 5: Server UI

# Software Architecture

Afbeelding met tekst, schermopname, cirkel, diagram

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schets, tekening, diagram

Automatisch gegenereerde beschrijving

Figure 6: Car and Controller Software Architecture

The car and controller software architecture is primarily managed by FreeRTOS. The tasks are configured with varying stack sizes, with the MQTT and BLE tasks allocated significantly more stack space compared to others. The MQTT task has the highest priority, ensuring uninterrupted data communication with the HiveMQ broker and enabling the server to continuously update gyroscope and accelerometer information. The BLE task holds the second-highest priority and facilitates communication of the car's speed while receiving joystick inputs to control the DC motors. Finally, the Speedometer and MotorDriver tasks share equal priority, handling their respective responsibilities efficiently.

The controller's software architecture consists of three tasks. The BLE task is assigned the highest priority and largest stack size. The other two tasks, SevenSegmentsDriver and JoystickData, are configured with equal lower priority and smaller stack sizes. The BLE task manages the transmission of joystick data to the car and receives the car's speed via BLE. Based on the received speed data, the SevenSegmentsDriver task updates the display to show the current speed. Meanwhile, the JoystickData task reads the ADC values from the X and Y pins, preparing the data for transmission through the BLE communication channel.

# Power Consumption

Following table summarizes power consumption of the car:

|  |  |  |  |
| --- | --- | --- | --- |
| State | Voltage (V) | Current (mA) | Power (W) |
| Idle | 12 | 175 | 2.1 |
| Full drive | 12 | 600 | 7.2 |
| Full drive low power | 12 | 400 | 4.8 |

# Result

Here are some pictures of both subsystems.

