Embedded Systems Team Project Report

2024 – 06 – 14

Team members:

DABWAN RANA ABDULLAH SAEED (2021034184),
Pak Margarita (9065420240),
Liu Henry Arthur Zhongsan (9062820244)

Phase 1: Route Memorization

In the context of our code, the route memorization phase is **not explicitly defined**. However, the robot’s movement and decision-making process can be considered as a form of route memorization.

Reason for failure:

Choosing an appropriate algorithm for route memorization was a challenging task. The algorithm needed to be efficient and reliable, capable of accurately capturing the layout of the track based on the sensor inputs. Furthermore, it needed to be robust enough to handle variations in the track and sensor inaccuracies.

In our case, a specific route memorization algorithm was not implemented. This was due to the complexity involved in designing and implementing such an algorithm. Integrating the route memorization algorithm with the movement control in Phase 2 added another layer of complexity. The robot needed to not only remember the route but also use this information to make decisions about its movement.

Phase 2: One-Brush Drawing

The second phase is implemented in the moveForward(), moveBackward(), rotate_Right(), rotate_Left(), and cancelRotate() functions.

The robot uses Pulse Width Modulation (PWM) to control the speed of its motors. The Move() function sets the duty cycle for the PWM signal, which in turn controls the speed of the motors. The Left_Forward(), Left_Backward(), Right_Forward(), and Right_Backward() functions control the direction of the motors.

The robot’s movement is adjusted based on the sensor inputs. **For example**, if the robot detects that it has deviated from the track, it will adjust its movement to get back on track. This is done by rotating the robot to the left or right or moving it forward or backward.

## Functions

### `PWM_Init34(uint16_t period, uint16_t duty3, uint16_t duty4)`

This function initializes the PWM signals for the robot's motors. The `period` parameter sets the period of the PWM signal, and `duty3` and `duty4` set the duty cycle for each motor.

### `Switch_Init(void)`

This function initializes the start switch of the robot.

### `WaitForStart(void)`

This function makes the robot wait until the start switch is pressed.

### `Motor_Init()`

This function initializes the robot's motors.

### `PWM_Duty3(uint16_t duty3)` and `PWM_Duty4(uint16_t duty4)`

These functions set the duty cycle for each motor, controlling the speed of the motors.

### `Move(uint16_t leftDuty, uint16_t rightDuty)`

This function controls the speed of the robot's motors. It sets the duty cycle for the PWM signal, which in turn controls the speed of the motors.

### `Left_Forward()`, `Left_Backward()`, `Right_Forward()`, `Right_Backward()`

These functions control the direction of the motors. They set the direction of the left and right motors to either forward or backward.

### `moveForward(x,y,delay)`, `moveBackward(x,y,delay)`, `rotate_Right(x,y,delay)`, `rotate_Left(x,y,delay)`

These functions control the robot's movement. They call the `Move()` function to set the speed of the motors and the direction functions to set the direction of the motors. They also include a delay to control the duration of the movement.

### `cancelRotate(delay)`

This function stops the robot's rotation. It sets the speed of the motors to 0 and includes a delay to control the duration of the stop.

### `IR_Init()`

This function initializes the IR sensors of the robot.

### `trackMovement()`

This function is the main control loop for the robot's movement. It reads the sensor inputs and calls the appropriate movement functions based on these inputs. It also includes logic to handle different scenarios such as when the robot deviates from the track or reaches a node.

## Main Function

The `main()` function initializes the clock, motors, IR sensors, and start switch. It then waits for the start switch to be pressed before calling the `trackMovement()` function to start the robot's movement.

Differences Between Expected and Actual Results

The expected result in Phase 2 is that the robot moves smoothly along the track, adjusting its movement as necessary based on the sensor inputs. However, the actual result differed due to various factors such as sensor inaccuracies, mechanical issues, or software bugs.

For example, the robot did not move as smoothly as expected, or did not adjust its movement correctly when it deviated from the track.