University of West Florida

Hal Marcus College of Science and Engineering

Autonomous Car Project

EEL 4744 – Microprocessor Applications

Gabriel J. Black

Spring 2025

Introduction

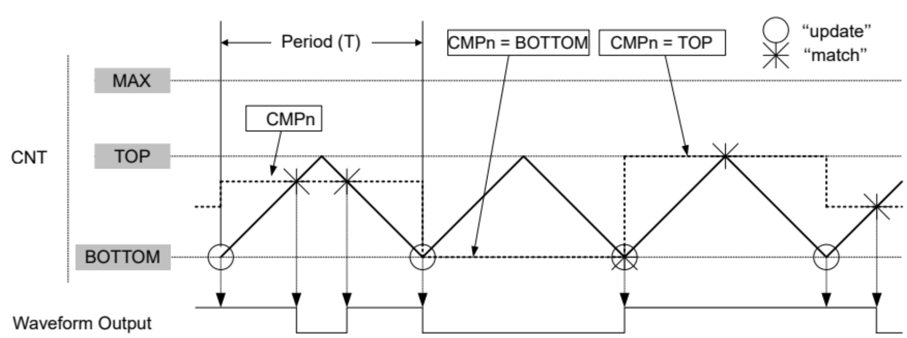
The goal of the autonomous car project was to design and construct an autonomous car using two different spatial sensors: a reflectance sensor and an ultrasonic sensor. The car is supposed to be able to follow a line of tape using the reflectance sensor and avoid obstacles using the ultrasonic sensor. The project involved many other parts, such as two motors, a motor driver, and the chassis used to construct the car. The AVR64DU32 microcontroller would control the car, and the code developed would be written in AVR assembly. This project engages students in developing basic artificial intelligence to control motors and sensors. In this report, I will focus specifically on the implementation of the line-following functionality using the reflectance sensor.

Background

Autonomous vehicles rely on sensors and algorithms to interpret their environment and navigate the world around them. The development of this simple autonomous car utilizes mainly conditional logic (branching in AVR assembly) to interpret the data from the sensors and move the car accordingly. I will now explain some of the techniques used to control the motor's speed and the reflectance sensors' data interpretation.

- ***DUAL-SLOPE PWM***

Dual-slope PWM generation uses the PER register to set the TOP value for the counter. It also uses CMP register 2 to control the duty cycle of the signal.



This diagram shows dual-slope (phase-correct) PWM operation, where the timer counts up and down between BOTTOM and TOP values. The output toggles at compare match points, resulting in a symmetric PWM signal with consistent high and low times each period. This technique is how the motor speed is controlled.

- ***EQUATIONS***

The following equations were used to generate the waveforms.

$$F_{SIGNAL} = \frac{f_{CLK_{PER}}}{2 * TOP * prescaler}$$

Alternatively,

$$(1)\ \ TOP = \frac{f_{CLK_{PER}}}{2 * F_{SIGNAL} * prescaler}$$

Equation (1) will be used to determine the required top for the period register for the chosen signal frequency.

$$Duty\ cycle = \frac{CMP}{TOP} * 100$$

Alternatively,

$$(2)\ MotorSpeed = CMP = \frac{(TOP * Duty\ Cycle)}{100}$$

Equation (2) will be used to determine the motor speed using a predetermined duty cycle.

- ***MPLAB x IDE***

MPLAB x IDE is an integrated development environment developed by Microchip Technology for programming and debugging PIC and AVR microcontrollers. MPLAB x IDE was the only editor used to develop this project. The editor features many debugging and data visualization plugins to help with the testing and design of the project. The data visualization was used to debug the program when testing the reflectance sensor.

- ***AVR Assembly***

The AVR Assembly language, as the name suggests, is a low-level programming language used to directly control AVR microcontrollers. It was designed for the AVR architecture's 8-bit RISC design. AVR assembly offers full control over registers, I/O ports, and memory, which is important for performance-critical tasks like motor control and sensor utilization.

- ***AVR64DU32 Microcontroller***

The AVR64DU32 is an 8-bit microcontroller developed by Microchip Technology. It features 64 KB of flash memory, 4 KB of SRAM, an up to 24 MHz clock, an on-board debugger for programming and testing, and up to 25 programmable GPIO pins. This microcontroller serves as the main control unit for the project and is essential to its functionality.

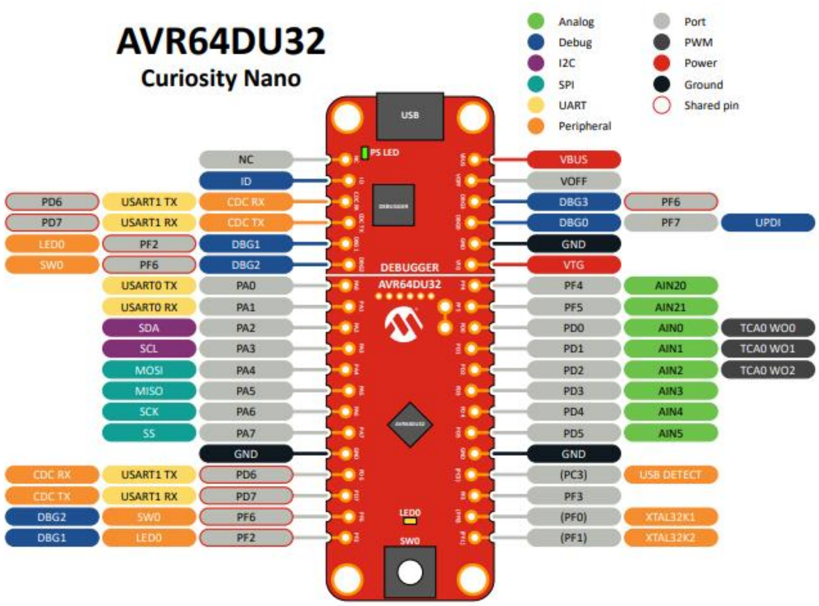


Figure 1: Pinout of AVR64DU32

- ***Motor Driver (TB6612)***

The TB6612 motor driver is capable of controlling two independent motors. Each motor is managed with two direction inputs and one PWM input for speed control, requiring a total of six digital signals. The driver receives logic-level power from the microcontroller through the VCC pin and motor power through the VM pin, which supports a voltage range from 4.5V to 13.5V. In this project, a 9V battery was used to supply motor power.

Figure 2: Motor driver.

- ***Reflectance Sensor Array (QTR-8A)***

The reflectance sensor features 8 LED/photoresistor pairs spaced approximately 0.375 inches apart, with each sensor providing an independent digital output. It requires a minimum supply voltage of 3.3V. For this project, the middle four sensors were used and connected to the lower four bits of Port A. When a sensor is placed over a non-reflective surface, the internal comparator is triggered, producing a high output (logic 1).

Figure 3: QTR-8A Reflectance sensor.

Code Structure

In this section, I will break down the code, how it was developed, and what each chunk of code does. Below you will find a chunk of code, then a short explanation of what it does.

```
; effectly a store immediate instruction
.MACRO STI
LDI R25, @0
STS @1, R25
.ENDMACRO

;calculations for Dual-Slope PWM 50Hz signal with 50% duty cycle
.EQU CLK_FREQ = 4000000
.EQU SIGNAL_FREQ = 50
.EQU PRESCALER = 1
.EQU TOP_VALUE = CLK_FREQ / (2 * SIGNAL_FREQ * PRESCALER)


; CMP_VALUE = MOTORSPEED
.EQU MOTORSPEED_100 = (TOP_VALUE * 100) / 100
.EQU MOTORSPEED_75 = (TOP_VALUE * 75) / 100
.EQU MOTORSPEED_50 = (TOP_VALUE * 50) / 100
.EQU MOTORSPEED_40 = (TOP_VALUE * 40) / 100
.EQU MOTORSPEED_25 = (TOP_VALUE * 25) / 100
.EQU MOTORSPEED_20 = (TOP_VALUE * 20) / 100
.EQU MOTORSPEED_10 = (TOP_VALUE * 10) / 100
```

Figure 4: Code directives.

This section of the code includes a macro called STI (Store Immediate) for loading 8 bits into memory using the STS instruction. This section also features the calculations for the top and motor speed (Duty cycle) of the PWM signals. The calculations are equations (1) and (2). They both utilize the .EQU directive to assign an equation to a variable.

```
MAIN:
;------------------------- PWM Motor Setup ----------------------------
;set dual slope pwm mode
STI 0b01110101, TCA0_SINGLE_CTRLB

;set port d as output for wave generators 0x03
;port f for testing 0x05
STI 0x03, PORTMUX_TCAROUTEA

;prescaler set to 1
STI 0x01, TCA0_SINGLE_CTRLA

;set new top
STI HIGH(TOP_VALUE), TCA0_SINGLE_PERH
STI LOW(TOP_VALUE), TCA0_SINGLE_PERL

;set 1st motor speed
STI HIGH(MOTORSPEED_75), TCA0_SINGLE_CMP2H
STI LOW(MOTORSPEED_75), TCA0_SINGLE_CMP2L

;set 2nd motor speed
STI HIGH(MOTORSPEED_75), TCA0_SINGLE_CMP1H
STI LOW(MOTORSPEED_75), TCA0_SINGLE_CMP1L


;set motor speed to portF
STI 0b00000110 , PORTD_DIR

;set portd as output
STI 0xF0 , PORTA_DIR

;set direction outputs
STI 0b01100000, PORTA_OUT
```

Figure 5: PWM Motor setup code.

At the beginning of the main label in the code, the PWM motor setup is configured. The code initializes a dual-slope PWM signal with a frequency of 50 Hz. The TCA0_SINGLE_CTRLB register sets both the waveform generation mode (in the lower 4 bits) and the enabled compare channels (in the upper 3 bits). It is configured for dual-slope PWM and enables outputs on CMP0, CMP1, and CMP2. The PORTMUX_TCAROUTEA register is set to 0x03 to route the PWM outputs to Port D. The prescaler is set to 1. The PER (Period) high and low registers define the top value of the timer using equation (1) to determine the PWM frequency. Initial motor speeds are defined in the CMP1 and CMP2 high and low byte registers. The Port D direction register sets PD1 and PD2 as outputs for the PWM signals. Additionally, the upper four bits of Port A are configured as outputs to control motor direction—these are hardcoded and not dynamically changed during operation.

```
;------------------------- Autonomous logic ---------------------------------
loop:
LDS R20, PORTA_IN
ANDI R20, 0x0F       ;and "0x0F" with incoming data to mask upper 4 bits
CPI R20, 0x06        ;compare with 0x06
BREQ GO_STRAIGHT     ;go straight if equal (z = 1)
BRGE TURN_RIGHT      ;turn right if R20 > 0x06
BRLT TURN_LEFT       ;turn lef if R20 < 0x06

rjmp loop
```

Figure 6: Automation logic.

The code in Figure 6 contains the main loop, which implements the core automation logic for the project. It begins by reading the reflectance sensor data through Port A pins A3–A0. To isolate the relevant sensor input, the data is ANDed with 0x0F, masking out the upper four bits and ensuring cleaner comparisons. The resulting value is then compared to 0x06, which corresponds to the condition where the sensor is aligned with the center of the electrical tape. If this condition is met, the Z (zero) flag is set, and the BREQ (Branch if Equal) instruction is executed, instructing the car to continue moving straight. If the condition is not met, the program checks whether a right turn is needed using the BRGE (Branch if Greater or Equal) instruction. Similarly, if the value is less than 0x06, the BRLT (Branch if Less Than) instruction is triggered to initiate a left turn.

```
GO_STRAIGHT:
 ;set 1st motor speed
STI HIGH(MOTORSPEED_40), TCA0_SINGLE_CMP2H
STI LOW(MOTORSPEED_40), TCA0_SINGLE_CMP2L

;set 2nd motor speed
STI HIGH(MOTORSPEED_40), TCA0_SINGLE_CMP1H
STI LOW(MOTORSPEED_40), TCA0_SINGLE_CMP1L

RJMP loop


TURN_RIGHT:

;set 1st motor speed
STI HIGH(MOTORSPEED_25), TCA0_SINGLE_CMP2H
STI LOW(MOTORSPEED_25), TCA0_SINGLE_CMP2L

;set 2nd motor speed
STI HIGH(MOTORSPEED_10), TCA0_SINGLE_CMP1H
STI LOW(MOTORSPEED_10), TCA0_SINGLE_CMP1L

RJMP loop



TURN_LEFT:

    ;set 1st motor speed
STI HIGH(MOTORSPEED_10), TCA0_SINGLE_CMP2H
STI LOW(MOTORSPEED_10), TCA0_SINGLE_CMP2L

;set 2nd motor speed
STI HIGH(MOTORSPEED_25), TCA0_SINGLE_CMP1H
STI LOW(MOTORSPEED_25), TCA0_SINGLE_CMP1L

RJMP loop
```

Figure 7: turning code logic.

The code in Figure 7 includes three labeled sections that control the motor speeds during turning maneuvers. Each label adjusts the PWM duty cycles to change the speed of the motors accordingly. For a right turn, the speed of the right motor is reduced while the left motor maintains its speed, causing the car to pivot right. The same logic applies for a left turn, where the left motor's speed is lowered to initiate the turn.

Schematic

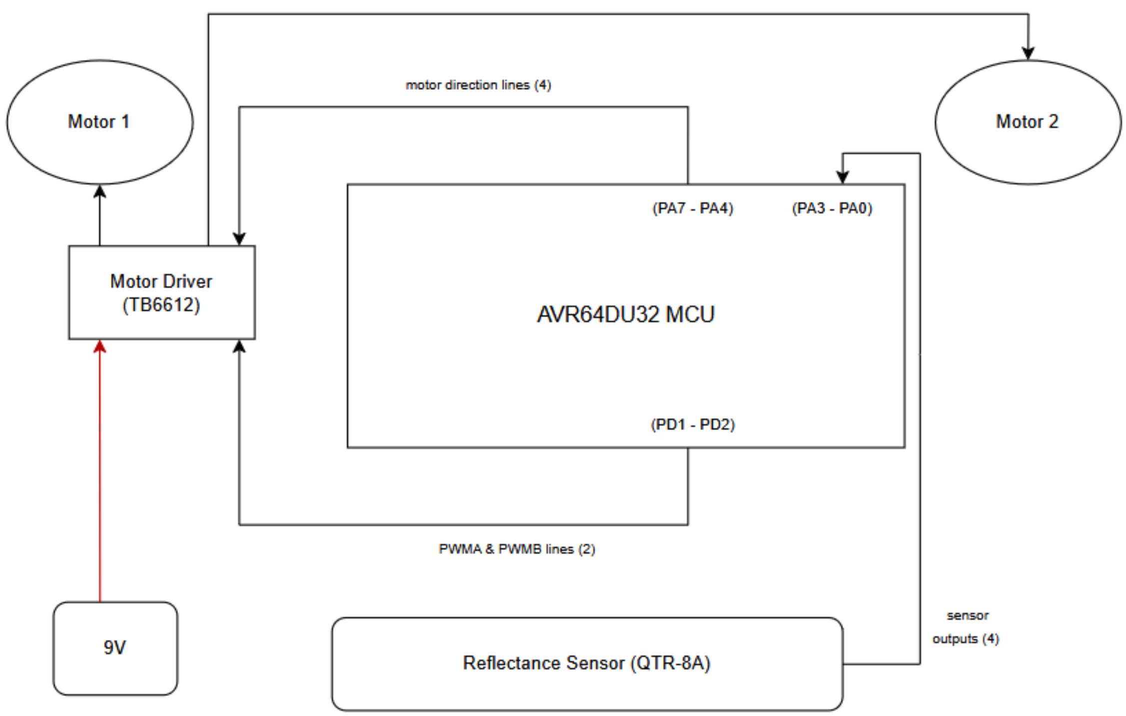


Figure 8: Schematic for autonomous car.

The schematic above shows the basic layout of the project and the most important connections for functional operation. The reflectance sensor is connected to ports A3–A0 as input data. The project requires a motor driver to supply enough current to the motors for proper operation. The motor driver has six inputs from the microcontroller, ports A7-A4 are the direction lines for motors. Ports D1 and D2 are the outputs of the PWM signals A and B. A 9V battery is needed to supply enough current to the motors (the microcontroller alone does not have enough current to drive the motors).

Results

I successfully designed and implemented an autonomous car capable of line following using a reflectance sensor. The car completed a full lap around a track made of electrical tape, accurately handling both left and right turns. To ensure smooth and stable navigation, the motor speeds were carefully calibrated through testing. One improvement for future iterations would be to increase the distance between the reflectance sensor and the surface to prevent false readings caused by surface irregularities.

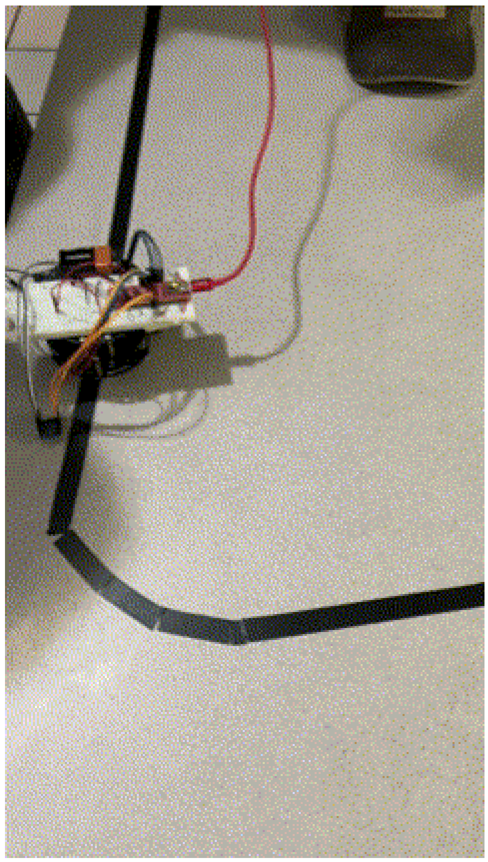


Figure 9: Gif of successful robot run.

Conclusion

This project successfully demonstrated the design and implementation of an autonomous car capable of line following using a reflectance sensor and AVR assembly programming. By integrating PWM-based motor control with sensor-driven decision-making, the car was able to navigate a taped track and handle directional changes with reliability. The use of the AVR64DU32 microcontroller provided precise control over the motors and sensor inputs, and MPLAB X IDE enabled efficient development and debugging. While only the line-following functionality was implemented in this version, the modular design allows for future expansion, including obstacle avoidance via ultrasonic sensing. Overall, the project was a practical application of embedded systems principles and low-level programming for autonomous navigation.

References

Microchip Technology Inc., *AVR64DU28/32 Preliminary Data Sheet*, DS40002548A, 2023. [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/AVR64DU_28_32_Prelim_DataSheet_DS40002548.pdf

Pololu Corporation, "QTR-8RC Reflectance Sensor Array," [Online]. Available: https://www.pololu.com/product/960

Adafruit Industries, “Adafruit TB6612 1.2A DC/Stepper Motor Driver Breakout Board,” Product ID: 2448. [Online]. Available: https://www.adafruit.com/product/2448

Microchip Technology Inc., *AVR64DU32 Curiosity Nano User Guide*, Document DS50003671A, 2024. [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/BoardDesignFiles/AVR64DU32-Curiosity-Nano-Design-Documentation.zip

Microchip Technology Inc., *AVR Instruction Set Manual*, Document DS40002198A, 2020. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/AVR-Instruction-Set-Manual-DS40002198A.pdf