Digital Design

Shreyas Kumar*
shreyas.kumar@icloud.com

Abstract
This is the final 8-week report. During the first 4 weeks, a series of basic projects were completed. These projects involved digital design using Arduino, an introduction to the PlatformIO IDE, assembly-level programming on the ATmega328P microcontroller, and Embedded C programming using AVR-GCC. In the last 4 weeks, two projects were undertaken. The objective of the first project was to enable a UGV to follow an exact straight path, while the second project focused on utilizing machine learning for beacon tracking using an unmanned ground vehicle (UGV) and a WiFi-enabled microcontroller such as the ESP32. The write-up for every project work has been arranged chronologically in this report. The general format for every write-up is Introduction, Working Theory, Circuit Diagram/Truth Tables/Boolean Logic/Digital Schematic/Components (if applicable), and Source Codes.

*The author was a Summer Research Fellow under Dr. G. V. V. Sharma, EED, IIT Hyderabad in the year 2023

Contents

1 Digital Design using Arduino

1.1 Introduction

The aim of this project is to provide a comprehensive understanding of Digital Design using Arduino and the PlatformIO IDE. The training program involved a set of basic projects that were implemented on a breadboard using an Arduino, a seven-segment display, and the relevant decoder IC. These experiments, when performed in a sequential manner, introduce the fundamental concepts of Elementary C programming and Digital Logic Design.

1.2 Working Theory

In this project, we use the 7474 D-Flip Flop ICs in a sequential circuit to realize a decade counter using Arduino and 7447 Decoder.

Arduino is a microcontroller which can be used to implement a variety of truth tables using its pins and ports as inputs and outputs. The boolean logic can also be implemented using K-maps or directly using loops in C programming. The code is written in PlatformIO IDE and the microcontroller is then programmed.

The decoder IC being used here is 7447. It translates a 4-bit input into outputs relevant to producing numeric patterns on the seven segment display.

The 7474 D-Flip Flops ICs are used to latch the output and send it back as input to the Arduino forming a feedback loop to produce the next decoding sequence by the Arduino as per the boolean logic.

1.3 Truth Table

1.3.1 For Seven Segment Display

The truth table for connecting Seven Segment Display is shown in Table 1.

INPUT				**OUTPUT**						
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0

Table 1: For Seven Segment Display connected directly to the Arduino microcontroller.

1.3.2 For Decoder IC 7447

The truth table for decoder IC input from the Arduino is shown in Table 2.

INPUT				OUTPUT			
Z	Y	X	W	D	C	B	A
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0

Table 2: For 7447 decoder IC, W, X, Y, Z are internal inputs for the microcontroller and A, B, C, D are corresponding outputs of the Arduino as inputs to the decoder IC.

1.4 Source Codes

1.4.1 With Decoder IC 7447

https://github.com/shreyas/FWC/blob/main/IDE/Decade%20Counter/with7447.cpp

1.4.2 Without Decoder IC 7447

https://github.com/shreyas/FWC/blob/main/IDE/Decade%20Counter/without7447.cpp

1.4.3 With D-Flip Flops

https://github.com/shreyas/FWC/blob/main/IDE/Decade%20Counter/withDFlipFlop.cpp

2 Boolean Verification using PlatformIO

2.1 Introduction

A Boolean function F of three variables X, Y, and Z is given as
F(X, Y, Z) = (X' + Y + Z).(X + Y' + Z').(X' + Y + Z').(X' Y' Z' + X' Y Z' + X Y Z')
which can be simplified to F(X, Y, Z) = (X' Z') + (Y Z'). The objective is to verify the truth table using Arduino in PlatformIO IDE.

2.2 Working Theory

The given function was simplified using Boolean Algebra as follows:

F(X, Y, Z) = (X' + Y + Z).(X + Y' + Z').(X' + Y + Z').(X' Y' Z' + X' Y Z' + X Y Z')
= (X' + Y + Z).(X + Y' + Z').(X' + Y + Z').(X' Z'.(Y + Y') + X Y Z')
= (X' + Y + Z).(X + Y' + Z').(X' + Y + Z').(X' Z' + X Y Z')
= (X' + Y + Z).(X + Y' + Z').(X' + Y + Z').(Z').(X' + X Y)
= (X' + Y + Z).(X + Y' + Z').(X' + Y + Z').(Z').(X' + Y)
= (X' + Y + Z).(X + Y' + Z').(X' + Y + Z').(Z').(X' + Y)

Let (X' + Y) = A

F(X, Y, Z) = (A + Z).(X + Y' + Z').(A + Z').(Z').(X' + Y)
= (X + Y' + Z').(A + ZZ').(Z').(X' + Y)
= (X + Y' + Z').(X' + Y).(Z').(X' + Y)
= (X + Y' + Z').(X' + Y).(Z')
= (X' + Y).(XZ' + Z'Y' + Z'Z')
= (X' + Y).(XZ' + Z'Y' + Z'Z')
= (X' + Y).(XZ' + Z'Y' + Z')
= (X' + Y).(Z').(X + Y' + 1)
= (Z').(XX' + X'Y' + X' + YX + YY' + Y)
= (Z').(X'Y' + X' + YX + Y)
= (Z').(X'.(Y' + 1) + Y.(X + 1))
= (Z').(X' + Y)

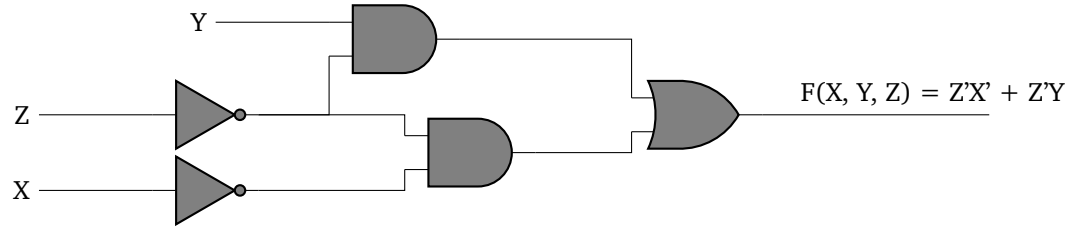
F(X, Y, Z) = Z'X' + Z'Y

2.3 Truth Table

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Table 3: Truth table for Boolean function 'F'

2.4 Digital Schematic



2.5 Components

Component	Values	Quantity
Arduino	UNO	1
Jumper Wires	M-M	10
Breadboard		1
LED		2
Resistor	220 ohms	1

Table 4: List of items required

2.6 Implementation and Source Code

Arduino PIN	INPUT	OUTPUT
2	X	
3	Y	
4	Z	
13		F

Table 5: Connections

Procedure:

a. Connect the circuit as per the above table.

b. Connect the output pin to LED.

c. Connect inputs to Vcc for logic 1, ground for logic 0.

d. Execute the circuit using the below code.

https://github.com/shr-eyas/FWC/blob/main/platformio.cpp

e. Change the values of X, Y, Z in the code and verify the truth table.

3 Assembly Programming on ATmega328P

3.1 Introduction

A decade counter was implemented by programming an Arduino using the Assembly language instructions for the ATmega328P. This enables one to know about the microcontroller architecture and assembly language.

3.2 Working Theory

In addition to what has been mentioned in Section 1.2, Assembly programming is specific to the microcontroller or microprocessor being used and hence is platform dependent. The assembly instructions are written in an IDE called Geany. Here the Assembly code targets the ATmega328P microcontroller on the Arduino board.

3.3 Truth Table

All the Truth Tables and Boolean Logics are same as mentioned in Section 1.3.

3.4 Source Codes

3.4.1 With Decoder IC 7447

https://github.com/shr-eyas/FWC/blob/main/Assembly/Decade%20Counter/with7447.asm

3.4.2 Without Decoder IC 7447

https://github.com/shr-eyas/FWC/blob/main/Assembly/Decade%20Counter/without7447.asm

3.4.3 With D-Flip Flops

https://github.com/shr-eyas/FWC/blob/main/Assembly/Decade%20Counter/withDFlipFlop.asm

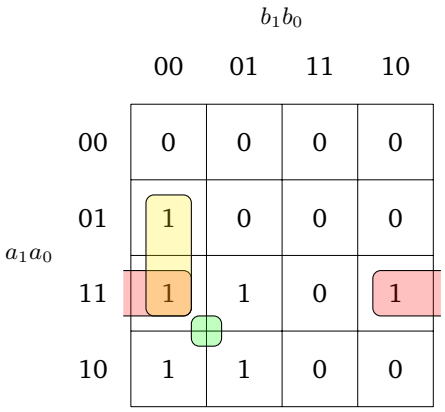
4 Boolean Verification in Assembly

4.1 Introduction

$A = a_1a_0$ and $B = b_1b_0$ are two 2-bit unsigned binary numbers. If $F(a_1, a_0, b_1, b_0)$ is a Boolean function such that $F = 1$ only when $A > B$, and $F = 0$ otherwise, then F can be minimized to the form:
$a_1\bar{b}_1 + a_1a_0\bar{b}_0 + a_0\bar{b}_0\bar{b}_1$.
Objective is to verify this equation on Arduino using Assembly Level Language for programming.

4.2 Working Theory

Using the boolean logic output F can be expressed in terms of the inputs X,Y,Z with the help of the following Karnaugh map.



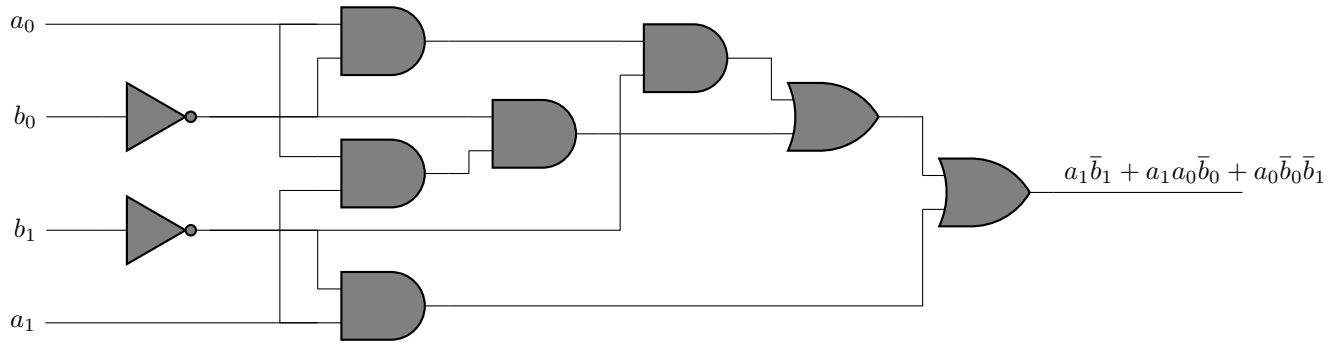
Karnaugh map simplification results in $a_1\bar{b}_1 + a_1a_0\bar{b}_0 + a_0\bar{b}_0\bar{b}_1$

4.3 Truth Table

a_1	a_0	b_1	b_0	**F**
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Table 6: Truth table for Boolean function 'F'

4.4 Digital Schematic



4.5 Components

Component	Values	Quantity
Arduino	UNO	1
Jumper Wires	M-M	10
Breadboard		1
LED		2
Resistor	220 ohms	1

Table 7: List of items required

4.6 Implementation and Source Code

Arduino PIN	INPUT	OUTPUT
2	a0	
3	a1	
4	b0	
5	b1	
8		F

Table 8: Connections

Procedure:

a. Connect the circuit as per the above table.

b. Connect the output pin to LED.

c. Connect inputs to Vcc for logic 1, ground for logic 0.

d. Execute the circuit using the below code.

https://github.com/shr-eyas/FWC/blob/main/Assembly/assembly.asm

e. Change the values of X, Y, Z in the code and verify the truth table.

5 Embedded C and AVR-GCC

5.1 Introduction

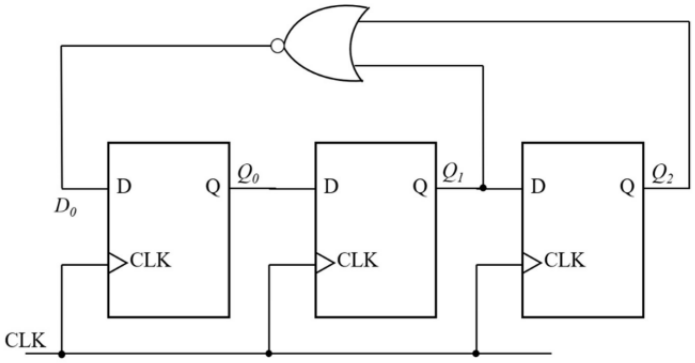
AVR-GCC, in particular, is a version of GCC optimized for compiling C and C++ code for AVR microcontrollers. Objective here is to verify the correct operation of a divide-by-5 counter implemented using a 7474 IC where the binary count is displayed using seven segment display. The program for verification must be written in Embedded C and compiled using AVR-GCC.

5.2 Components

Component	Values	Quantity
Arduino	UNO	1
Jumper Wires	M-M	30
Breadboard		1
Seven Segment Display	Common Anode	1
Resistor	220 ohms	1
IC	7474	2
IC	7447	1

Table 9: List of items required

5.3 Digital Schematic



5.3.1 Boolean Equation

From the circuit diagram we can conclude:

$$D0 = \overline{(Q1 + Q2)} \tag{1}$$

$$D1 = Q0 \tag{2}$$

$$D2 = Q1 \tag{3}$$

5.4 Truth Table

Q2	Q1	Q0	D2	D1	D0	Q2'	Q1'	Q0'
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	1
0	1	1	1	1	0	1	1	0
1	1	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0

Table 10: Truth table for the given circuit

7447	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$
Display	a	b	c	d	e	f	g

5.5 Implementation and Source Code

a. Make the connections between the seven segment display and the 7447 IC as shown below.

b. Connect the Arduino, 7447 and the two 7474 ICs according to below table.

	INPUT			OUTPUT			CLOCK		5V			
	Q0	Q1	Q2	Q0'	Q1'	Q2'						
Arduino	D6	D7	D8	D2	D3	D4	D13					
7474	5	9		2	12		CLK1	CLK2	1	4	10	13
7474			5			2	CLK1	CLK2	1	4	10	13
7447				7	1	2			16			

c. Hence we have implemented the divide by 5 counter digital circuit. Execute the circuit using below code.

https://github.com/shr-eyas/FWC/blob/main/Embedded%20C/counter.c

6 Project 1

6.1 Introduction

This main objective of this project is to make an unmanned ground vehicle (UGV) follow an exact straight path which it naturally does not do because of unavoidable mechanical errors.

6.2 Working Theory

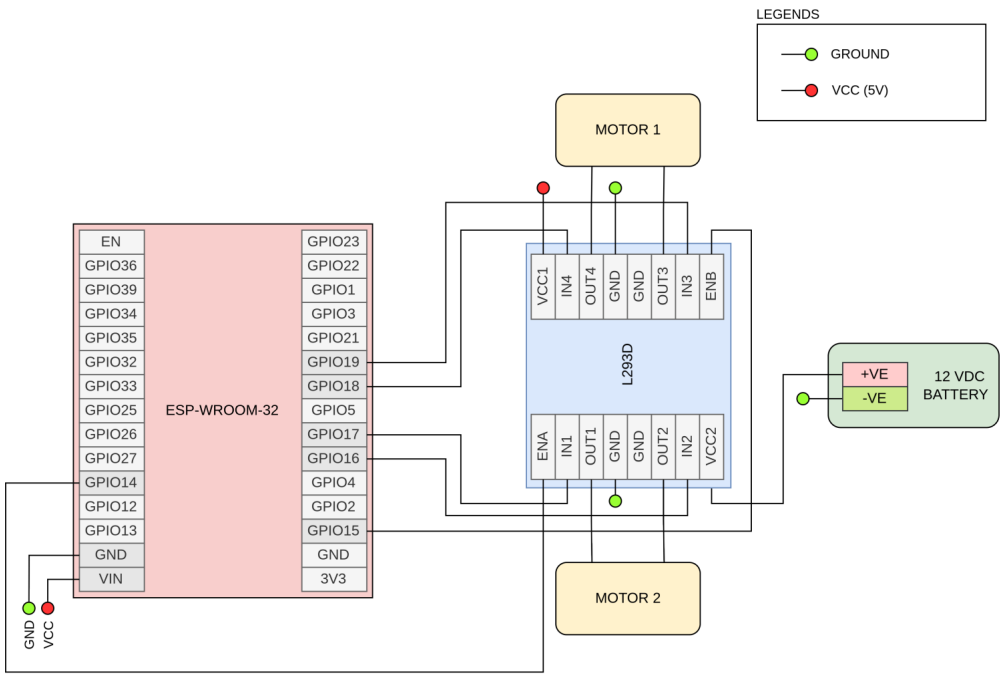
To achieve the objective, a simple technique known as differential drive control is used. Encoders attached to the wheels will count the number of rotations and hence distance traveled by each wheel. Further, a closed loop control system directs the wheel to control its rotation speed on basis of the error measured. Error here is the difference between number of rotations of each wheel. Ideally, both the wheels must rotate by equal amounts to make the UGV go straight.

6.3 Components

Component	Values	Quantity
ESP	32	1
Motor Driver	L293D	1
DC Motor	7.4V	2
UGV Chassis		1
Breadboard		1
Encoder	Optical	2
Jumper Wires	M-M	10
Jumper Wires	M-F	10
LiPo Battery	7.4V	1

Table 11: List of items required

6.4 Digital Schematic



6.5 Implementation and Source Code

a. Assemble the UGV Chassis, make the connections as per the wiring diagram and put both the systems together.

b. Upload the following code on ESP32. This code has no closed control loop and just provides both motors with equal power.

https://github.com/shr-eyas/Robotics/blob/main/IIT%20Hyderabad%20/UGV/noLoopUGV.cpp

c. Now put the UGV on the floor with no obstacles in the way and then plug in the battery.

d. The UGV deviates instead of going exactly straight. This deviation is easily visible as it mostly ends up going in a curved path.

e. Following this, re-upload the following code on ESP32. This code has a closed control loop which will make the UGV follow the path of a straight line.

https://github.com/shr-eyas/Robotics/blob/main/IIT%20Hyderabad%20/UGV/pdUGV.cpp

f. Now once again put the UGV on the floor with no obstacles in the way and then plug in the battery.

g. This time, it can be seen that the UGV follows a straight line path instead of going on a curved trajectory.

7 Project 2

7.1 Introduction

This main objective of this project is to demonstrate the use of machine learning in beacon tracking using an unmanned ground vehicle (UGV) and a WiFi-enabled microcontroller such as the ESP32.

7.2 Working Theory

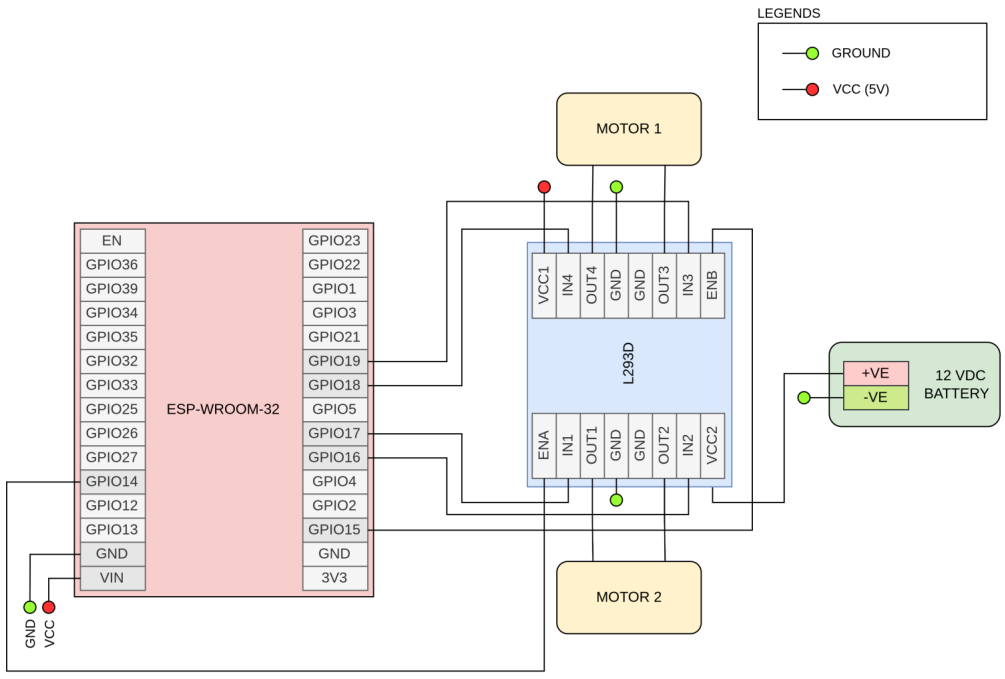
Initially, the ESP32 mounted on the car reads the RSSI (Radio Signal Strength Indicator) levels in the forward, right, and left directions through suitable in-place rotation. To ensure accurate RSSI readings, an average of 20 RSSI values is taken for each direction. Subsequently, the car rotates towards the direction with the highest RSSI level, indicating the strongest signal from the beacon. It then proceeds to move forward continuously in that direction, maintaining alignment with the beacon's signal. By repeating above steps again and again, the car navigates towards the beacon.

7.3 Components

Component	Values	Quantity
ESP	32	1
Motor Driver	L293D	1
DC Motor	7.4V	2
UGV Chassis		1
Breadboard		1
Jumper Wires	M-M	10
Jumper Wires	M-F	10
LiPo Battery	7.4V	1

Table 12: List of items required

7.4 Digital Schematic



7.5 Implementation and Source Code

a. Assemble the UGV Chassis, make the connections as per the wiring diagram and put both the systems together.

b. Upload the following code on ESP32.

https://github.com/shr-eyas/Robotics/blob/main/IIT%20Hyderabad%20/Beacon%20Tracking/beacon.cpp

c. Now put the phone at a reasonable distance from the UGV with no obstacles in the way and then turn on the hotspot. The UGV should travel towards the phone and stop near it.

d. The algorithm described above is very basic and crude way to navigate toward the beacon. Frequent left and right rotation is required to identity the direction having the maximum RSSI level.

e. Using this algorithm, only 60-70 percent of the time (i.e. 60-70% Accuracy), the car rotates and moves towards the correct direction. Due to low accuracy, The car may take time to reach till the beacon.