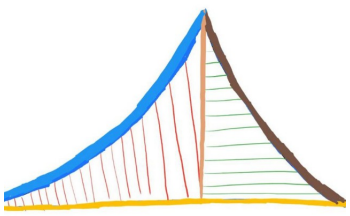
DIGITAL DESIGN

Through Arduino

G. V. V. Sharma



Contents

Introduction

This book introduces digital design through using the arduino framework.

Chapter 1

Installation

1.1. Termux

1. On your android device, install apkpure from

```
https://mapkpurecom/
```

2. Install Termux from apkpure

3. Install basic packages on termux

```
#Give termux access to your user directory in android
termux-setup-storage

#Upgrade packages
apt update && apt upgrade
apt install build-essential openssh

#Mandatory packages
apt install curl git wget subversion proot proot-distro python nmap neovim ranger
#--------------------End Install Termux
```

```
    ------------------------------
```

4. Install Ubuntu on termux

```
proot-distro install ubuntu
proot-distro login ubuntu
```

1.2. Platformio

1. Install Packages

```
apt update && apt upgrade
apt install apt-utils build-essential cmake neovim
apt install git wget subversion imagemagick nano
apt install avra avrdude gcc-avr avr-libc
#--------------------End Installing ubuntu on termux
    ------------------------------


#-------------------- Installing python3 on termuxubuntu
    ------------------------------
apt install python3-pip python3-numpy python3-scipy python3-matplotlib
    python3-mpmath python3-sympy python3-cvxopt
#-------------------- End installing python3 on termuxubuntu
    ------------------------------


#-------------------- Installing platformio on termuxubuntu
```

```
    ------------------------------
pip3 install platformio
#-------------------- End installing python3 on termuxubuntu
    ------------------------------
```

2. Execute the following on ubuntu

```
cd /sdcard/Download
svn co https://githubcom/gadepall/fwc-/trunk/ide/piosetup/codes
cd codes
pio run
```

3. Connect your arduino to the laptop/rpi and type

```
pio run -t nobuild -t upload
```

4. The LED beside pin 3 will start blinking

1.3. Arduino Droid

1. Install ArduinoDroid from apkpure

2. Open ArduinoDroid and grant all permissions

3. Connect the Arduino to your phone via USB-OTG

4. For flashing the bin files, in ArduinoDroid,

```
Actions->Upload->Upload Precompiled
```

then go to your working directory and select

```
pio/build/uno/firmwarehex
```

for uploading hex file to the Arduino Uno

5. The LED beside pin 3 will start blinking

Chapter 2

Seven Segment Display

We show how to control a seven segment display.

2.1. Components

Component	Value	Quantity
Breadboard		1
Resistor	$\geq 220\Omega$	1
Arduino	Uno	1
Seven Segment Display	Common Anode	1
Jumper Wires		20

Table 2.1:

2.1.1. Breadboard

The breadboard can be divided into 5 segments. In each of the green segements, the pins are internally connected so as to have the same voltage. Similarly, in the central segments, the pins in each column are internally connected in the same fashion as the blue columns.

2.1.2. Seven Segment Display

The seven segment display in Fig. 2.2 has eight pins, a, b, c, d, e, f, g and *dot* that take an active LOW input, i.e. the LED will glow only if the input is connected to ground. Each of these pins is connected to an LED segment. The *dot* pin is reserved for the $\cdot$ LED.

2.1.3. Arduino

The Arduino Uno has some ground pins, analog input pins A0-A3 and digital pins D1-D13 that can be used for both input as well as output. It also has two power pins that can generate $3.3V$ and $5V$. In the following exercises, only the GND, $5V$ and digital pins will be used.

2.2. Display Control through Hardware

2.2.1. Powering the Display

1. Plug the display to the breadboard in Fig. 2.1 and make the connections in Table 2.2. Henceforth, all 5V and GND connections will be made from the breadboard.

Arduino	**Breadboard**
5V	Top Green
GND	Bottom Green

Table 2.2:

2. Make the connections in Table 2.3.

3. Connect the Arduino to the computer. The DOT led should glow.

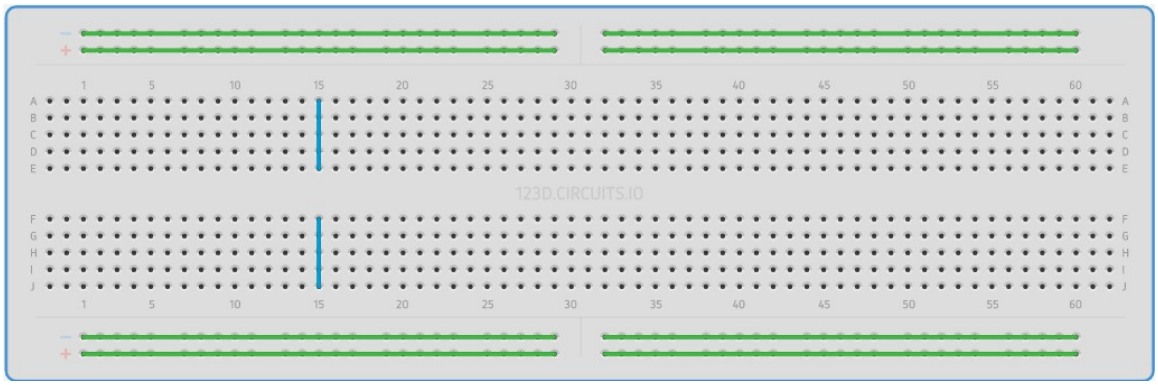


Figure 2.1:

Breadboard		**Display**
5V	Resistor	COM
GND		DOT

Table 2.3:

2.2.2. Controlling the Display

Fig. 2.3 explains how to get decimal digits using the seven segment display. GND=0.

1. Generate the number 1 on the display by connecting only the pins b and c to GND (=0). This corresponds to the first row of 2.4. 1 means not connecting to GND.

2. Repeat the above exercise to generate the number 2 on the display.

3. Draw the numbers 0-9 as in Fig. 2.3 and complete Table 2.4

a	**b**	**c**	**d**	**e**	**f**	**g**	**decimal**
0	0	0	0	0	0	1	0

Table 2.4:

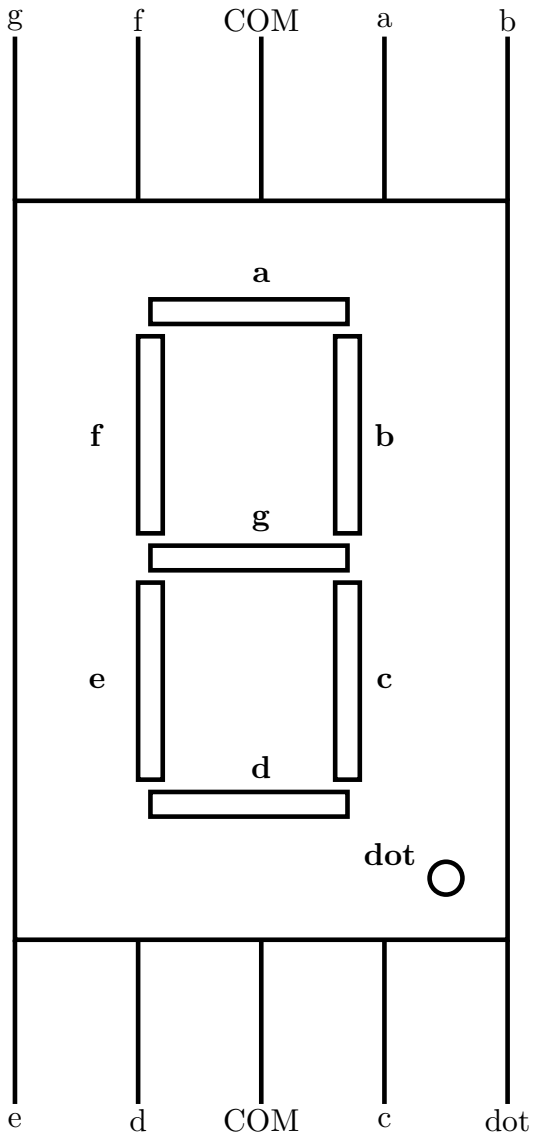


Figure 2.2:

2.3. Display Control through Software

1. Make connections according to Table 2.5

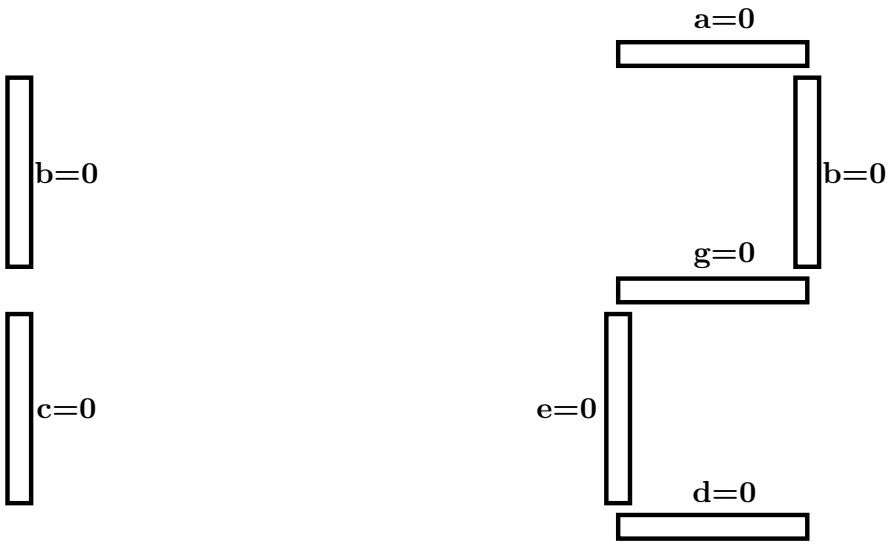


Figure 2.3:

Arduino	2	3	4	5	6	7	8
Display	a	b	c	d	e	f	g

Table 2.5:

2. Download the following code using the arduino IDE and execute

```
void sevenseg(int a,int b,int c,int d,int e,int f,int g)
{
  digitalWrite(2, a);
  digitalWrite(3, b);
  digitalWrite(4, c);
  digitalWrite(5, d);
  digitalWrite(6, e);
```

```
  digitalWrite(7, f);
  digitalWrite(8, g);
}
void setup()
{
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(7, OUTPUT);
    pinMode(8, OUTPUT);
}
void loop()
{
sevenseg(1,0,0,1,1,1,1);
}
```

3. Now generate the numbers 0-9 by modifying the above program.

Chapter 3

7447

Here we show how to use the 7447 BCD-Seven Segment Display decoder to learn Boolean logic.

3.1. Components

Component	Value	Quantity
Resistor	220 Ohm	1
Arduino	UNO	1
Seven Segment Display		1
Decoder	7447	1
Jumper Wires	M-M	20
Breadboard		1

Table 3.1:

3.2. Hardware

1. Make connections between the seven segment display in Fig. 2.2 and the 7447 IC in Fig. 3.1 as shown in Table 3.2

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

Table 3.2:

2. Make connections to the lower pins of the 7447 according to Table 3.3 and connect $V_{CC} = 5$V. You should see the number 0 displayed for 0000 and 1 for 0001.

D	**C**	**B**	**A**	**Decimal**
0	0	0	0	0
0	0	0	1	1

Table 3.3:

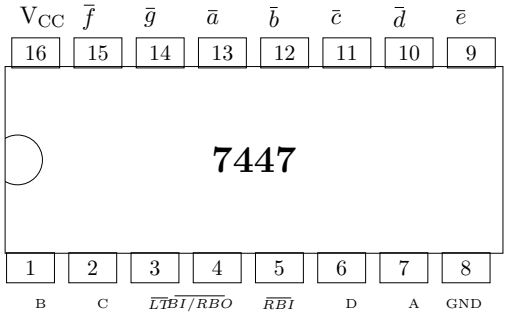


Figure 3.1:

3. Complete Table 3.3 by generating all numbers between 0-9.

3.3. Software

1. Now make the connections as per Table 3.4 and execute the following program after downloading

```
wget https://raw.githubusercontent.com/gadepall/arduino/master/7447/codes/
    gvv_ard_7447/gvv_ard_7447.ino
```

7447	D	C	B	A
Arduino	5	4	3	2

Table 3.4:

In the truth table in Table 3.5, W, X, Y, Z are the inputs and A, B, C, D are the outputs. This table represents the system that increments the numbers 0-8 by 1 and resets the number 9 to 0 Note that $D = 1$ for the inputs 0111 and 1000. Using boolean logic,

$$D = WXYZ^{'} + W^{'}X^{'}Y^{'}Z \tag{3.1}$$

Note that 0111 results in the expression $WXYZ^{'}$ and 1000 yields $W^{'}X^{'}Y^{'}Z$.

2. The code below realizes the Boolean logic for B, C and D in Table 3.5. Write the logic for A and verify.

```
//Declaring all variables as integers
int Z=0,Y=0,X=0,W=1;
int D,C,B,A;


//Code released under GNU GPL. Free to use for anything.
void disp_7447(int D, int C, int B, int A)
{
  digitalWrite(2, A); //LSB
  digitalWrite(3, B);
  digitalWrite(4, C);
```

```
  digitalWrite(5, D); //MSB


}
// the setup function runs once when you press reset or power the board
void setup() {
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
}


// the loop function runs over and over again forever
void loop() {
  A=0;
  B=(W&&!X&&!Y&&!Z) || (!W&&X&&!Y&&!Z) || (W&&!X&&Y&&!Z) || (!W
      &&X&&Y&&!Z);
  C=(W&&X&&!Y&&!Z) || (!W&&!X&&Y&&!Z) || (W&&!X&&Y&&!Z) || (!W&&
      X&&Y&&!Z);
  D = (W&&X&&Y&&!Z)||(!W&&!X&&!Y&&Z);


disp_7447(D,C,B,A);
}
//&& is the AND operation
// || is the OR operation
// ! is the NOT operation
```

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 3.5:

3. Now make additional connections as shown in Table 3.6 and execute the following code. Comment.

```
//Declaring all variables as integers
int Z,Y,X,W;


//Code released under GNU GPL. Free to use for anything.
void disp_7447(int D, int C, int B, int A)
{
  digitalWrite(2, A); //LSB
  digitalWrite(3, B);
  digitalWrite(4, C);
  digitalWrite(5, D); //MSB


}
// the setup function runs once when you press reset or power the board
```

```
void setup() {
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(6, INPUT);
    pinMode(7, INPUT);
    pinMode(8, INPUT);
    pinMode(9, INPUT);

}

// the loop function runs over and over again forever
void loop() {

W = digitalRead(6);//LSB
X = digitalRead(7);
Y = digitalRead(8);
Z = digitalRead(9);//MSB

disp_7447(Z,Y,X,W);
}
```

Solution: In this exercise, we are taking the number 5 as input to the arduino and displaying it on the seven segment display using the 7447 IC.

	Z	Y	X	W
Input	0	1	0	1
Arduino	9	8	7	6

Table 3.6:

4. Verify the above code for all inputs from 0-9.

5. Now write a program where

   (a) the binary inputs are given by connecting to 0 and 1 on the breadboard

   (b) incremented by 1 using Table 3.5 and

   (c) the incremented value is displayed on the seven segment display.

6. Write the truth table for the 7447 IC and obtain the corresponding boolean logic equations.

7. Implement the 7447 logic in the arudino. Verify that your arduino now behaves like the 7447 IC.

Chapter 4

Karnaugh Map

We explain Karnaugh maps (K-map) by finding the logic functions for the incrementing decoder

4.1. Incrementing Decoder

The incrementing decoder takes the numbers $0, , \ldots, 9$ in binary as inputs and generates the consecutive number as output The corresponding truth table is available in Table 4.1

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 4.1:

4.2. Karnaugh Map

Using Boolean logic, output A in Table 4.1 can be expressed in terms of the inputs W, X, Y, Z as

$$A = W'X'Y'Z' + W'XY'Z' + W'X'YZ' + W'XYZ' + W'X'Y'Z \quad (4.1)$$

1. K-Map for A: The expression in (4.1) can be minimized using the K-map in Fig 4.1 In Fig 4.1, the implicants in boxes 0,2,4,6 result in $W'Z'$ The implicants in boxes 0,8 result in $W'X'Y'$ Thus, after minimization using Fig 4.2, (4.1) can be expressed as

$$A = W'Z' + W'X'Y' \quad (4.2)$$

Using the fact that

$$\begin{aligned} X + X' &= \\ XX' &= 0, \end{aligned} \quad (4.3)$$

derive (4.2) from (4.1) algebraically

2. K-Map for B: From Table 4.1, using boolean logic,

$$B = WX'Y'Z' + W'XY'Z' + WX'YZ' + W'XYZ' \quad (4.4)$$

Show that (4.4) can be reduced to

$$B = WX'Z' + W'XZ' \quad (4.5)$$

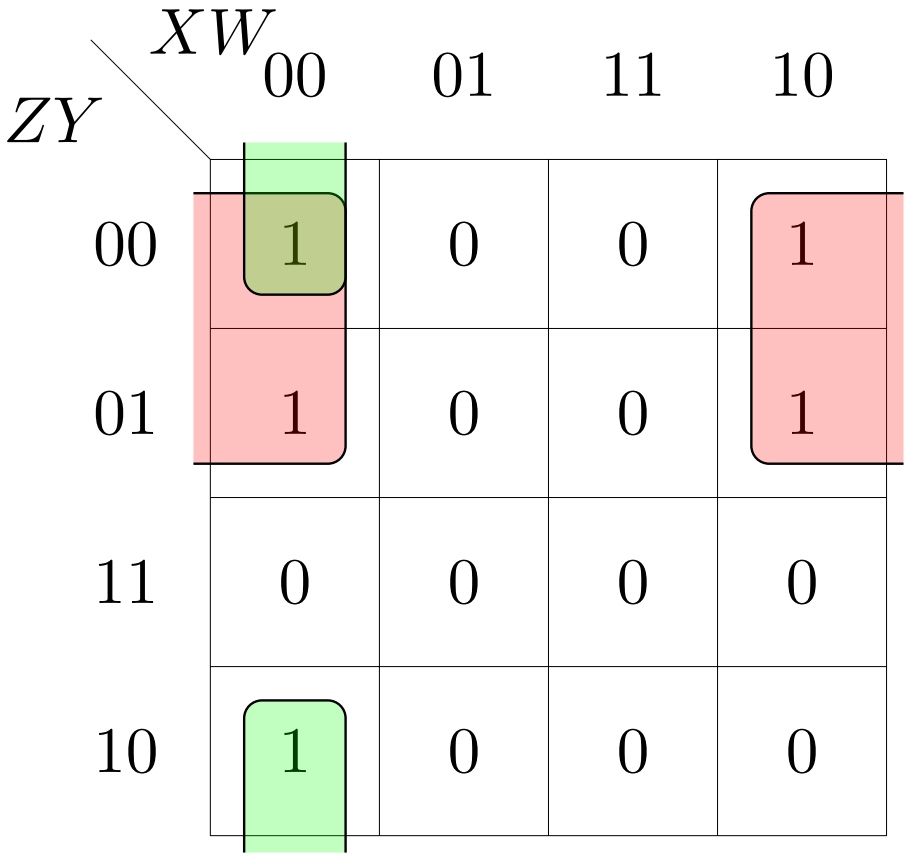


Figure 4.1: K-map for A

using Fig 4.2

3. Derive (4.5) from (4.4) algebraically using (4.3)

4. K-Map for C: From Table 4.1, using boolean logic,

$$C = WXY'Z' + W'X'YZ' + WX'YZ' + W'XYZ' \tag{4.6}$$

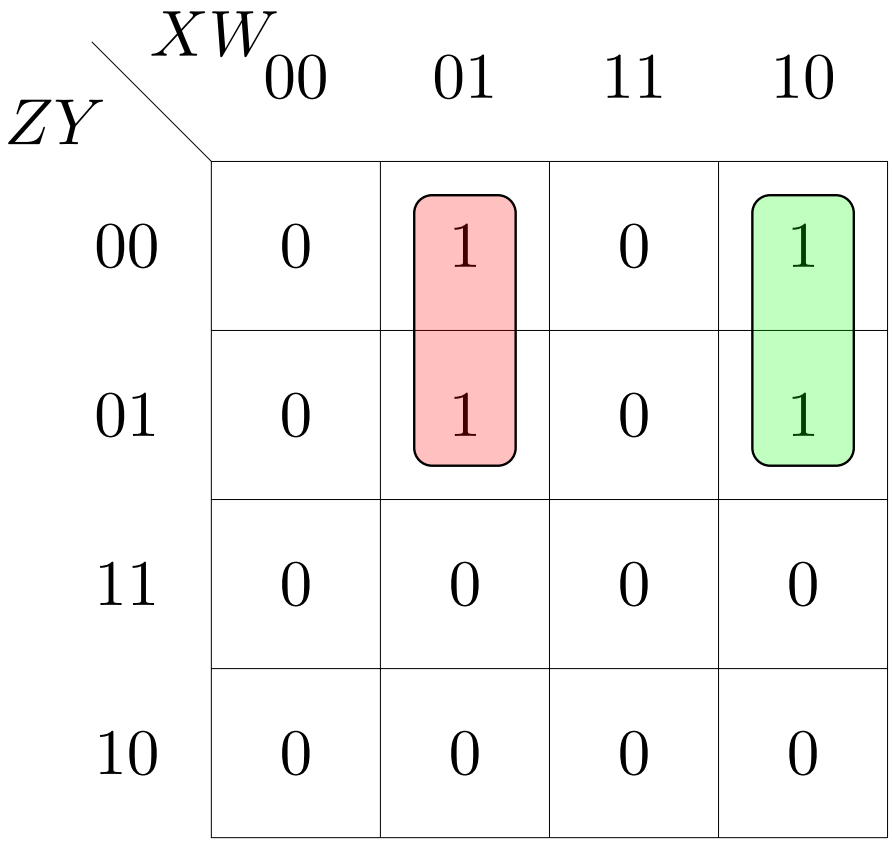


Figure 4.2: K-map for B

Show that (4.6) can be reduced to

$$C = WXY'Z' + X'YZ' + W'YZ' \tag{4.7}$$

using Fig 4.3

5. Derive (4.7) from (4.6) algebraically using (4.3)

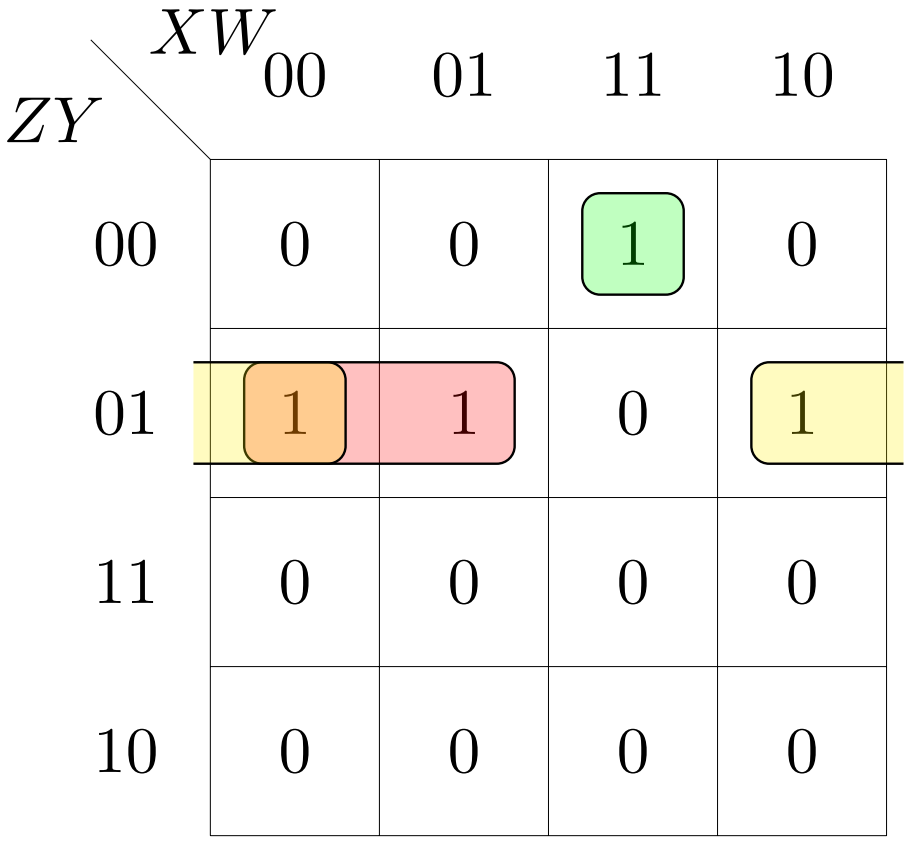


Figure 4.3: K-map for C

6. K-Map for D: From Table 4.1, using boolean logic,

$$D = WXYZ' + W'X'Y'Z \tag{4.8}$$

7. Minimize (4.8) using Fig 4.4

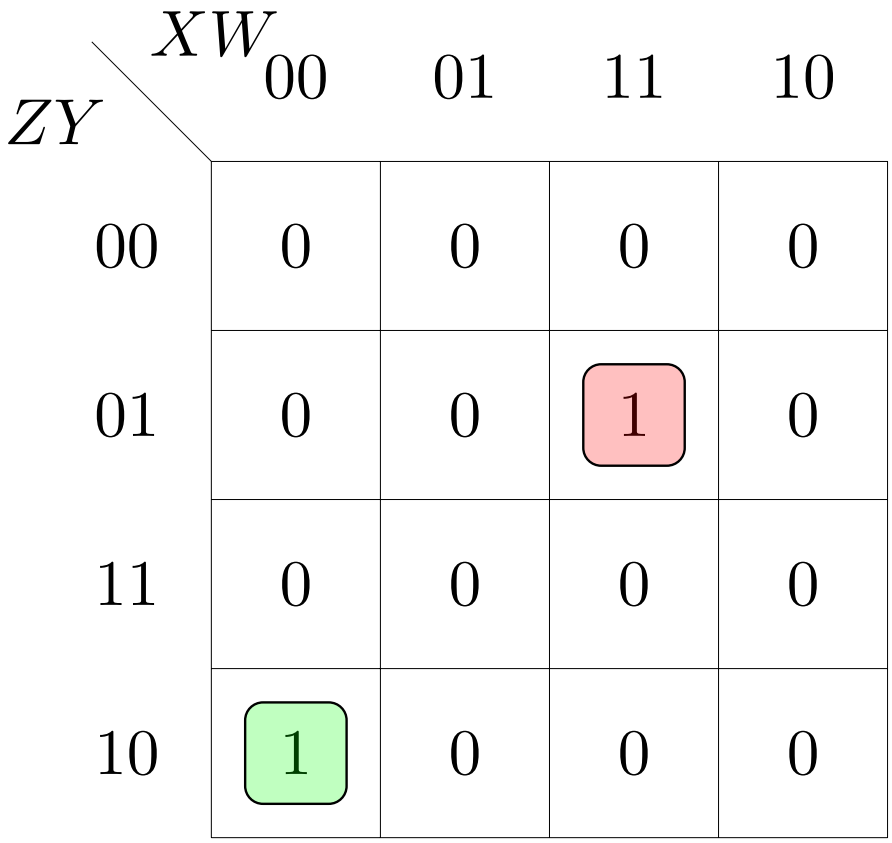


Figure 4.4: K-map for D

8. Download the code in

```
wget https://rawgithubusercontentcom/gadepall/arduino/master/7447/codes/
    inc_dec/inc_decino
```

and modify it using the K-Map equations for A,B,C and D Execute and verify

9. Display Decoder: Table 4.2 is the truth table for the display decoder in Fig. 3.1. Use

D	C	B	A	**a**	**b**	**c**	**d**	**e**	**f**	**g**	Decimal
0	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	0
0	0	0	1	**1**	**0**	**0**	**1**	**1**	**1**	**1**	1
0	0	1	0	**0**	**0**	**1**	**0**	**0**	**1**	**0**	2
0	0	1	1	**0**	**0**	**0**	**0**	**1**	**1**	**0**	3
0	1	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**0**	4
0	1	0	1	**0**	**1**	**0**	**0**	**1**	**0**	**0**	5
0	1	1	0	**0**	**1**	**0**	**0**	**0**	**0**	**0**	6
0	1	1	1	**0**	**0**	**0**	**1**	**1**	**1**	**1**	7
1	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	8
1	0	0	1	**0**	**0**	**0**	**1**	**1**	**0**	**0**	9

Table 4.2: Truth table for display decoder.

K-maps to obtain the minimized expressions for a, b, c, d, e, f, g in terms of A, B, C, D with and without don't care conditions

Chapter 5

Dont Care

We explain Karnaugh maps (K-map) using don't care conditions

5.1. Don't Care Conditions

1. Don't Care Conditions: 4 binary digits are used in the incrementing decoder in Table 4.1. However, only the numbers from 0-9 are used as input/output in the decoder and we don't care about the numbers from 0-5 This phenomenon can be addressed by revising the truth table in Table 4.1 to obtain Table 5.1

2. The revised K-map for A is available in Fig 5.1. Show that

$$A = W' \tag{5.1}$$

3. The revised K-map for B is available in Fig 5.2 Show that

$$B = WX'Z' + W'X \tag{5.2}$$

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**
1	0	1	0	**-**	**-**	**-**	**-**
1	0	1	1	**-**	**-**	**-**	**-**
1	1	0	0	**-**	**-**	**-**	**-**
1	1	0	1	**-**	**-**	**-**	**-**
1	1	1	0	**-**	**-**	**-**	**-**
1	1	1	1	**-**	**-**	**-**	**-**

Table 5.1:

4. The revised K-map for C is available in Fig 5.3 Show that

$$C = X'Y + W'Y + WXY' \tag{5.3}$$

5. The revised K-map for D is available in Fig 5.4 Show that

$$D = W'Z + WXY \tag{5.4}$$

6. Verify the incrementing decoder with don't care conditions using the arduino

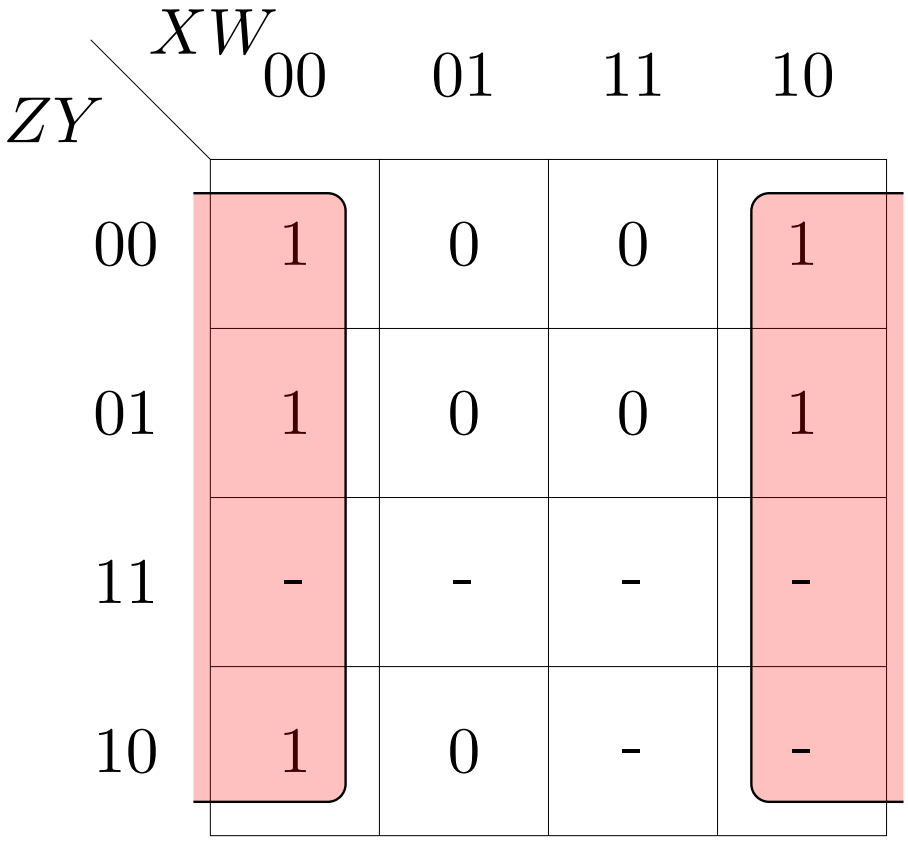


Figure 5.1: K-map for A with don't cares

7. Display Decoder: Use K-maps to obtain the minimized expressions for a, b, c, d, e, f, g in terms of A, B, C, D with don't care conditions

8. Verify the display decoder with don't care conditions using arduino

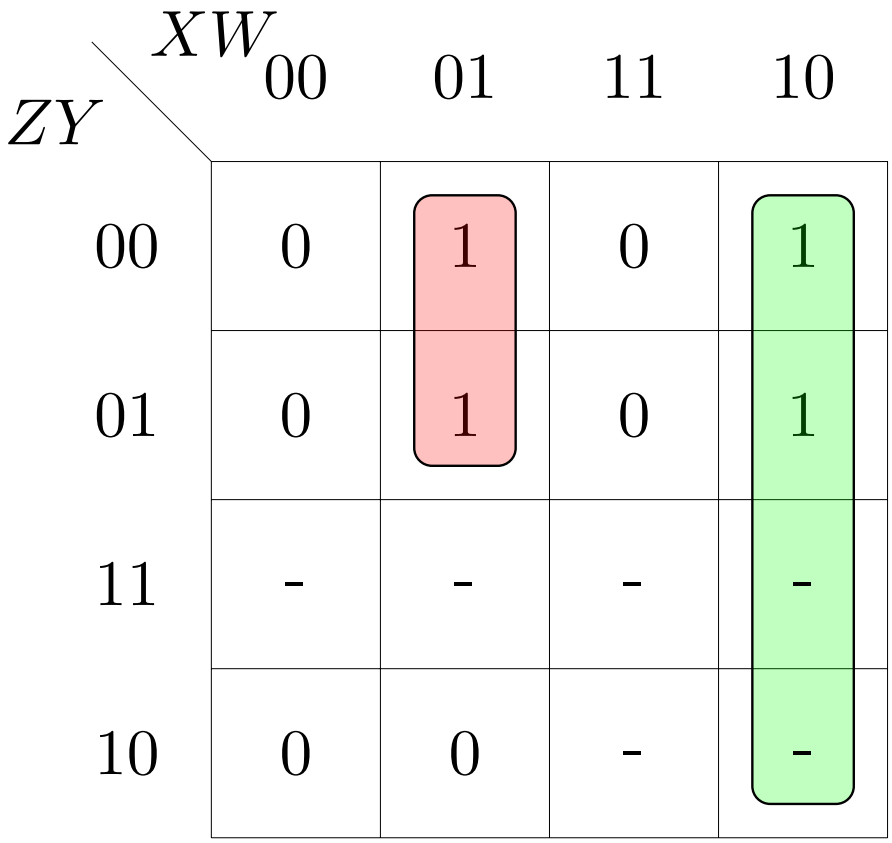


Figure 5.2: K-map for B with don't cares

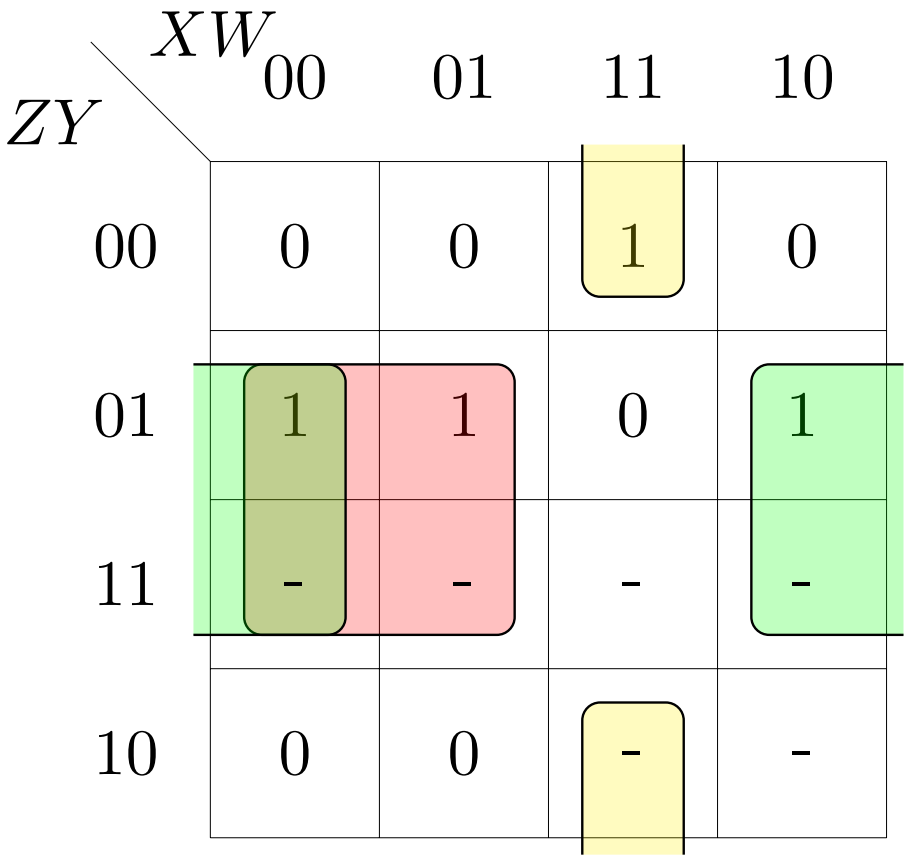


Figure 5.3: K-map for C with don't cares

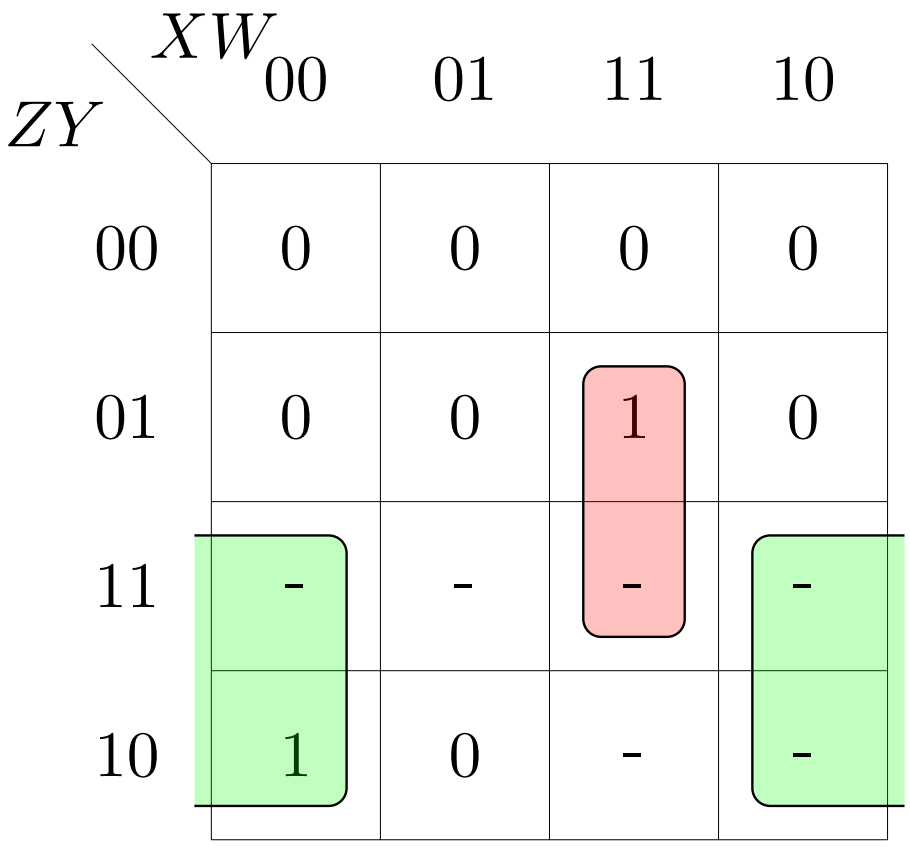


Figure 5.4: K-map for D with don't cares

Chapter 6

7474

We show how to use the 7474 D-Flip Flop ICs in a sequential circuit to realize a decade counter.

6.1. Components

Component	Value	Quantity
Breadboard		1
Resistor	$\geq 220\Omega$	1
Arduino	Uno	1
Seven Segment Display	Common Anode	1
Decoder	7447	1
Flip Flop	7474	2
Jumper Wires		20

Table 6.1:

6.2. Decade Counter

1. Generate the CLOCK signal using the **blink** program.

	INPUT				**OUTPUT**				CLOCK		5V			
	W	X	Y	Z	A	B	C	D						
Arduino	D6	D7	D8	D9	D2	D3	D4	D5	D13					
7474	5	9			2	12			CLK1	CLK2	1	4	10	13
7474			5	9			2	12	CLK1	CLK2	1	4	10	13
7447					7	1	2	6			16			

Table 6.2:

2. Connect the Arduino, 7447 and the two 7474 ICs according to Table 6.2 and Fig. 6.2.

   The pin diagram for 7474 is available in Fig. 6.1

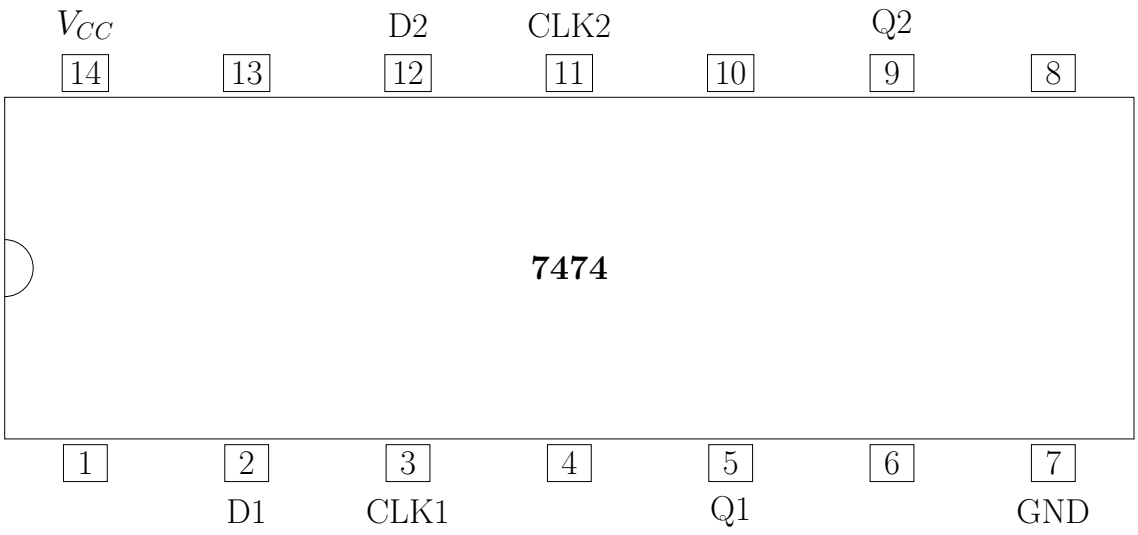


Figure 6.1:

3. Realize the decade counter in Fig. 6.2.

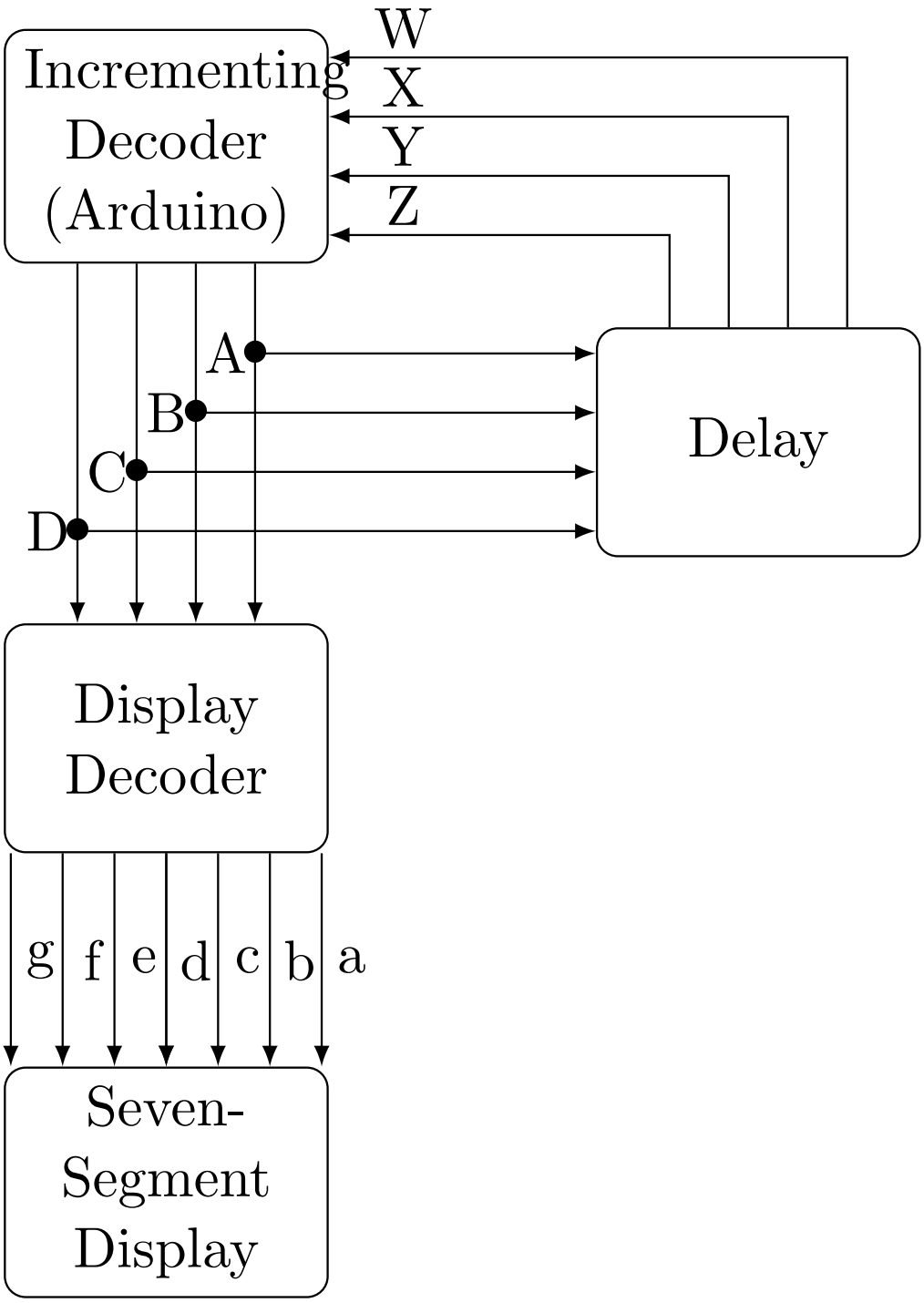

Incrementing
Decoder
(Arduino)
W
X
Y
Z
A
B
C
D
Delay
Display
Decoder
g
f
e
d
c
b
a
Seven-
Segment
Display

Figure 6.2: