Digital Logic

Dr TK Lam

Department of Computer Science and Engineering

香港中文大學
The Chinese University of Hong Kong

Overview

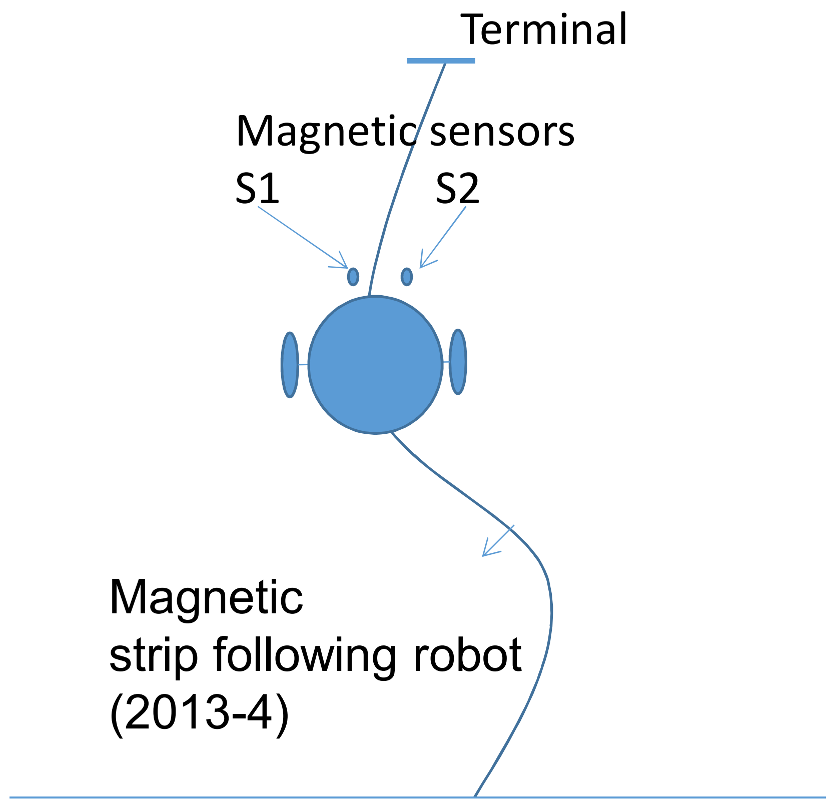
- 1. What is Digital Logic?
- 2. Digital Operations (AND, OR, NOT)
- 3. Truth Table
- 4. Software Implementation of Digital Operations

Motivations and Plans

- The brain of our robot is a set of digital logic functions
- We will introduce two techniques in digital logic design in this lecture
  - Logic formula
  - Truth table
- We will use a program in a micro-controller system to implement these techniques
  - Software implementation

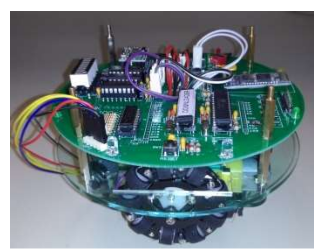
Example

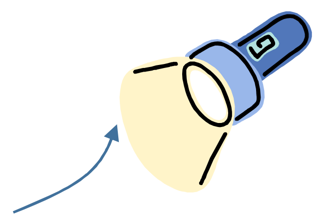
- How to keep the robot to move forward? Method:
  - If the robot deviates to the left, turn right
  - If the robot deviates to the right, turn left



Magnetic strip following robot (2013-4)

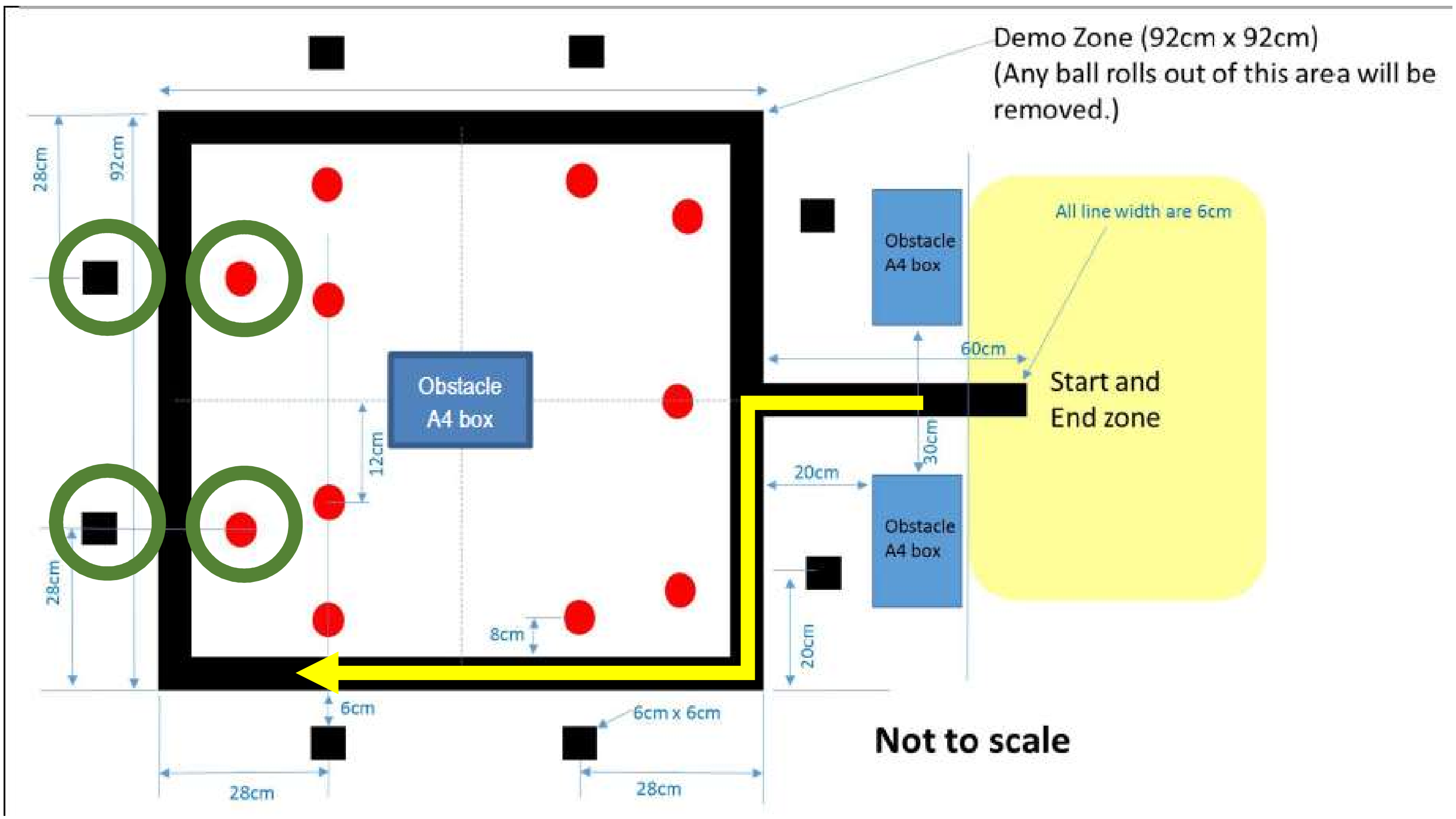
- The above rules are logic functions and operations.





Light following robot (2014-5)

2017-18



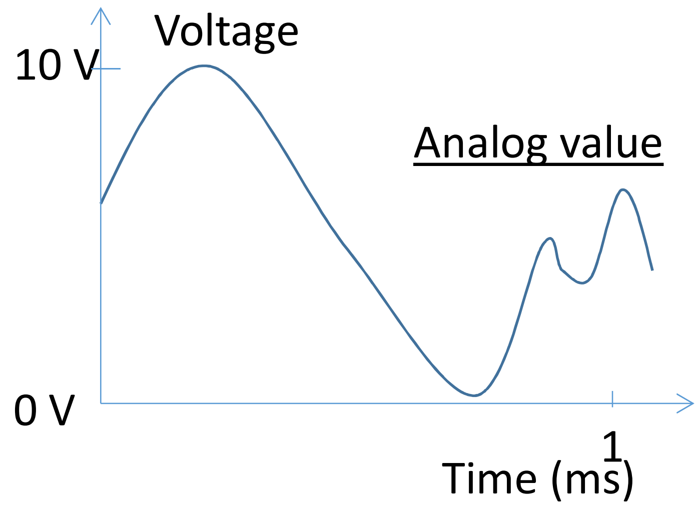
Source: 2017-18 ENGG1100 Project Specifications

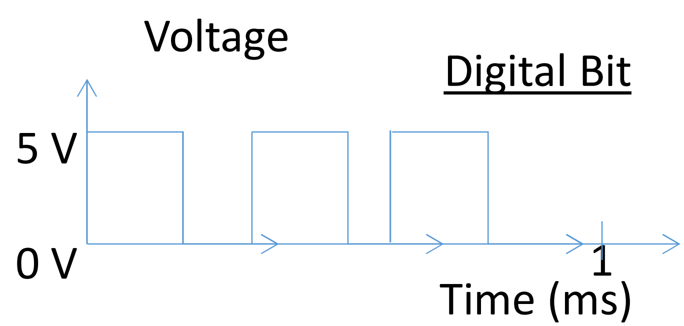
1. What is Digital Logic?

Understanding the difference between Digital and Analog operations

Analog and Digital Signals

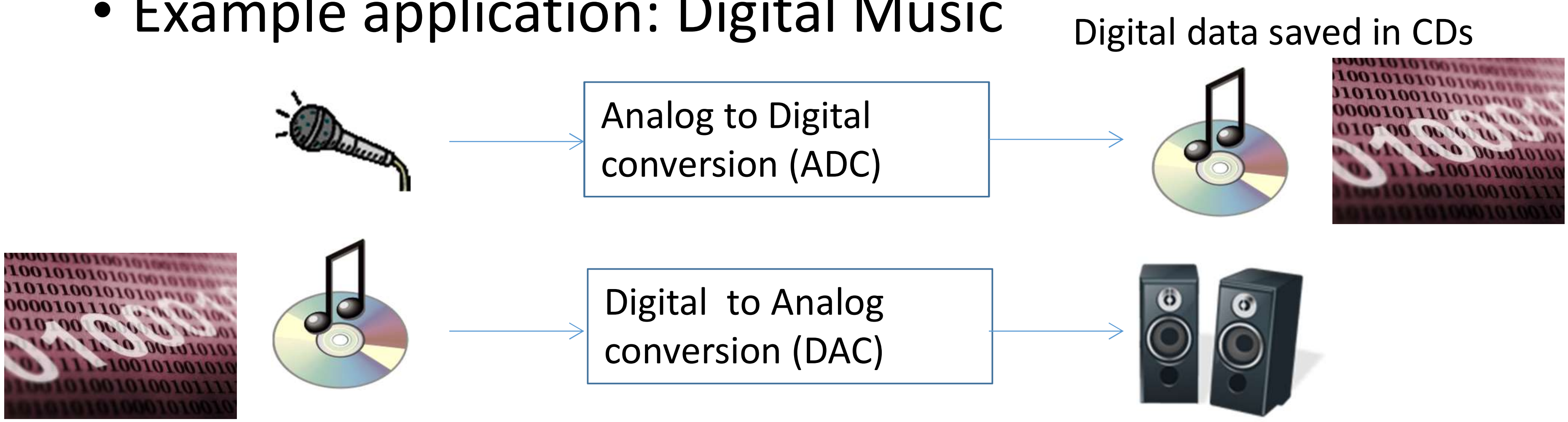
- Analog signals: the signal can be any values within the valid range
  - Example: Range = 0 ~ 10 Volts
  - The signal can be 1.356 Volts or 2.432 Volts
- Digital signals: It can only be HIGH (or called ‘1’) or LOW (or called ‘0’).
  - In TTL Transistor-transistor-logic standard:
    - High = Logic 1 = 5 Volts
    - Low = Logic 0 = 0 Volt
  - Usually several bits (more than one digital bit) are used to represent a number corresponding to an analog value
    - Example: 8-bit to represent a value from 0 to 2^8-1 = 255, or 32-bit to represent a value from 0 to 2^{32}-1





Why Digital Logic?

- A signal is represented by '1' or '0'
  - Easy to be implemented in electronic circuit
  - Less likely to be interfered by noise, temperature and radiation
  - Easy to store, and error correction
- Example application: Digital Music



Digital I/O Circuit

- Switch circuit as digital INPUT
  - Key pressed ➔ HIGH input
  - Key released ➔ LOW input
- LED circuit as digital OUTPUT
  - HIGH output ➔ LED on
  - LOW output ➔ LED off

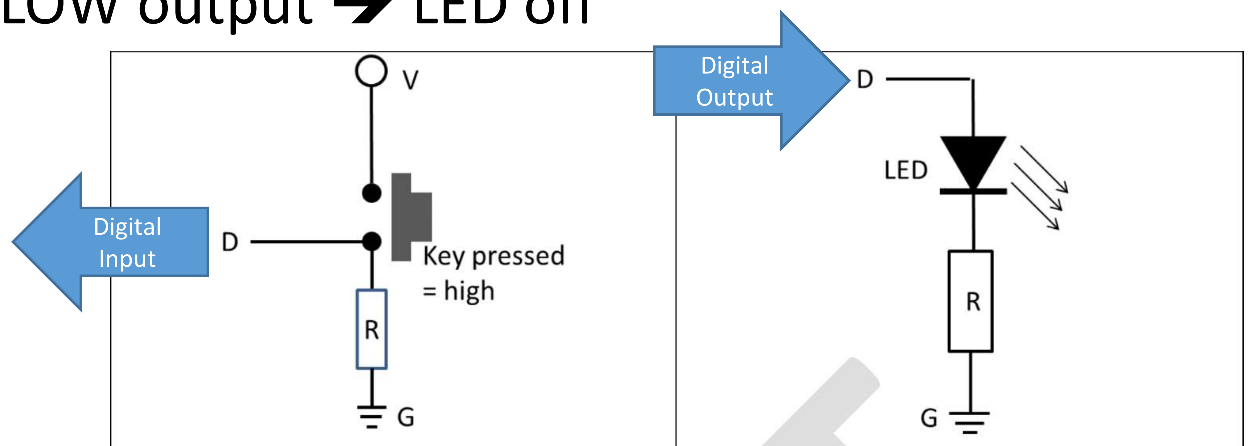


Figure-4.7: The switch circuit on the keyboard

Figure-4.8: The LED circuit on the keyboard

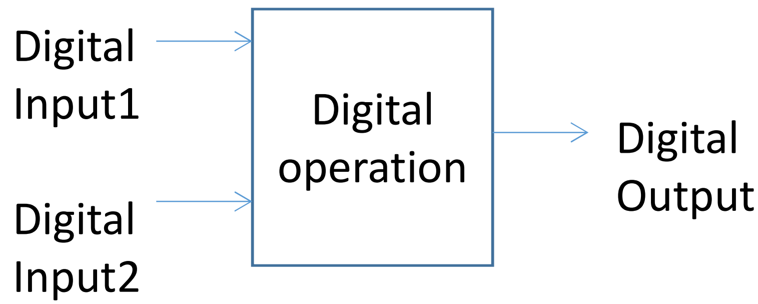
2. Digital Operations

AND, OR, NOT

Digital Operations

- Combine inputs to generate outputs
  - In arithmetic operations: 2 + 3 = 5
  - In digital operations: we need a truth table to see the result

- 3 popular digital operations you will learn here
  - AND
  - OR
  - NOT (Negation)

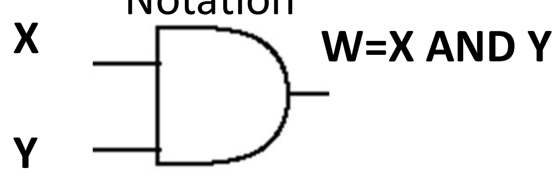


AND operation, example in real life

- You get a Degree from CUHK if you take 123 units and your GPA is greater than 1.5
  - (X=take 123 units) AND (Y=GPA>1.5) then you can get a Degree from CUHK (W)

- You must eat and drink in order to be alive
  - (X=eat) AND (Y=drink) then you can be alive (W)

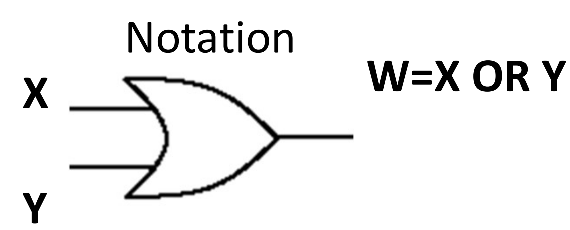
Notation



OR operation, example in real life

- If you live in Mongkok, you either take bus or train to the Chinese University
  - (X=take bus) or (Y=take train) then you can come to the University (W)

- You can ride on a bus if you pay by cash or pay using octopus
  - (X=pay by cash) or (Y=pay by octopus) then you can ride on the bus (W)

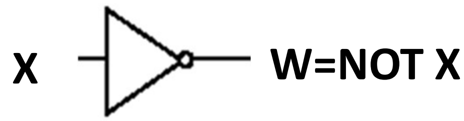


NOT operation, example in real life

- I don't love you = NOT (I love you)
  - NOT (X=I love you) means I don't love you (W)

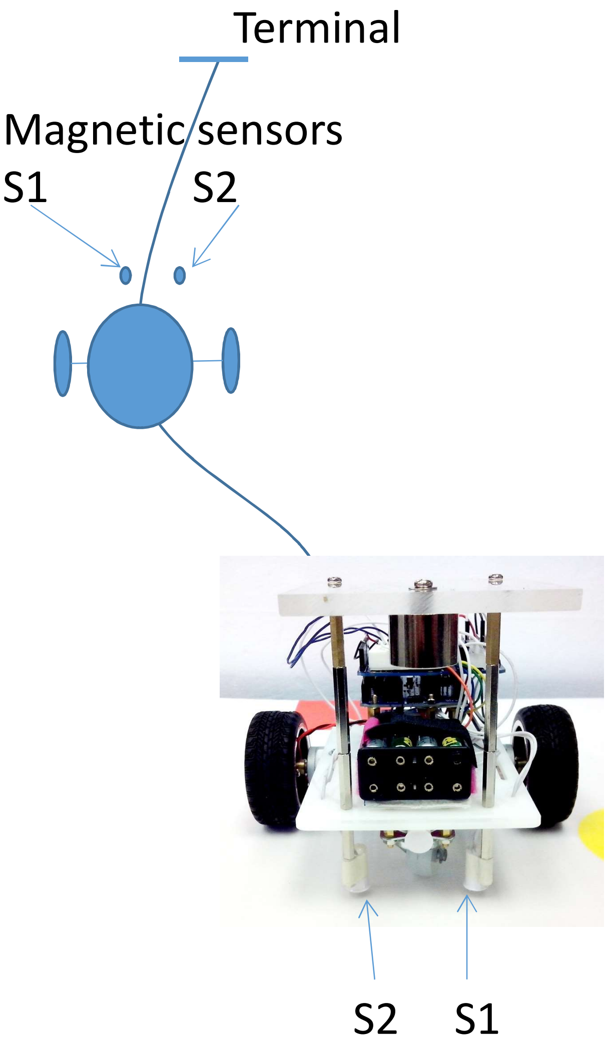
- You are not rich = NOT (You are rich)
  - NOT(X=you are rich) that means you are poor (W)

Notation

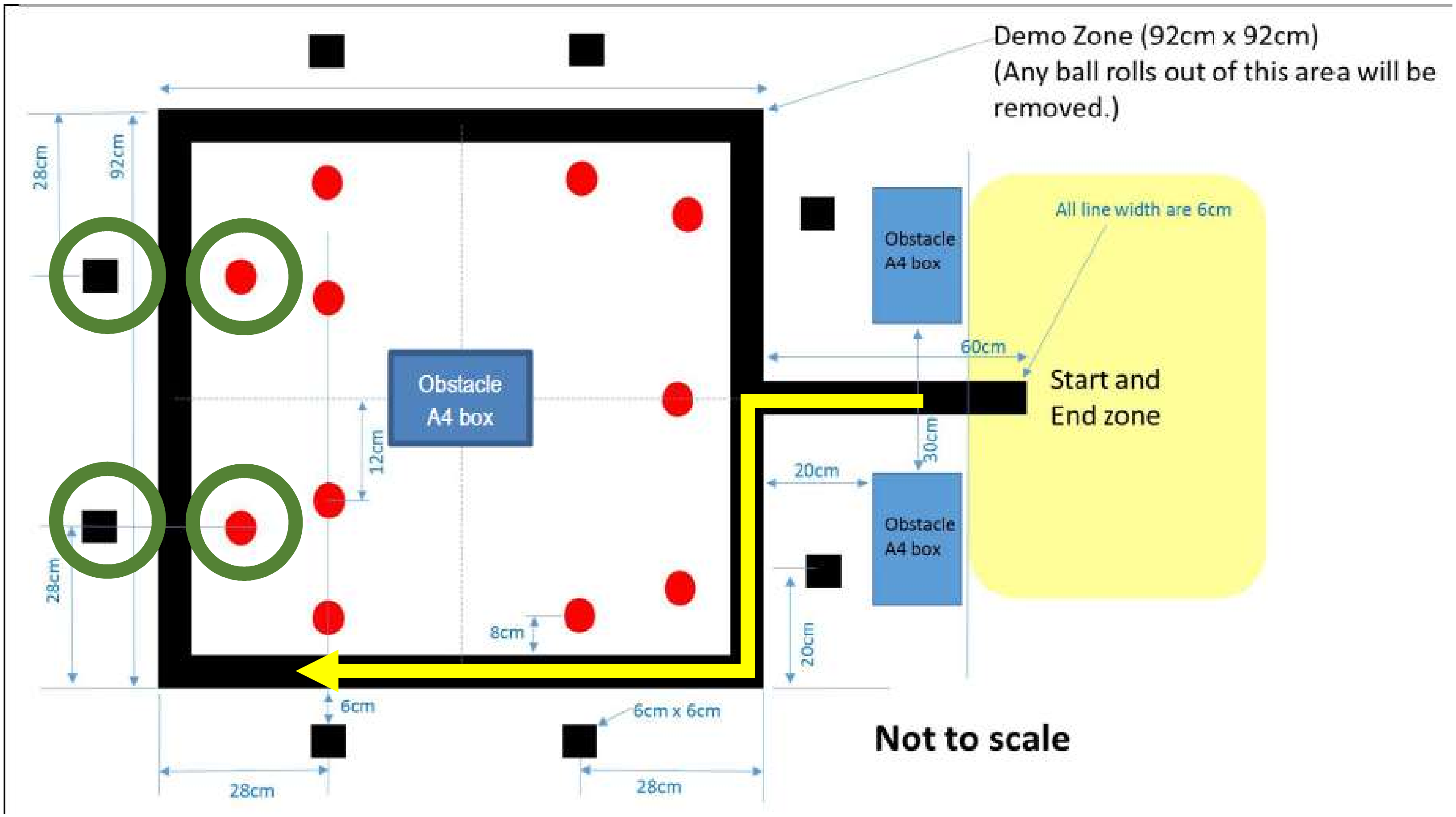


Example: Path Following Robot Control

- 2 sensors: S1 & S2
- If S2 detects the magnetic strip, but not S1, has the robot deviated to the right or left of the path?
- Ans.: Left



2017-18



Source: 2017-18 ENGG1100 Project Specifications

3. Truth table

A method to represent logic functions for digital signals

Truth table

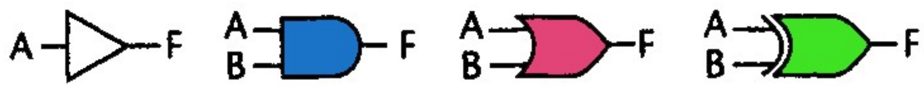
- The idea is to have all different combinations of inputs arranged in a table
- Each combination gives one output
- For n digital inputs , there will be 2^n different combinations. i.e. the truth table has 2^n rows
- Example:
  - n=2 (X and Y as inputs), so there are 2^n=4 rows
  - You can see that no two rows have the same combination of inputs
  - All possible combinations of the inputs (X,Y) are found in the truth table

- Example

Input: X	Input: Y	W= Output for the operation
0	0	?
0	1	?
1	0	?
1	1	?

? = depends on the operation

Common Logics and Truth Tables



Buffer $F = A$

A	F
0	0
1	1

AND $F = AB$

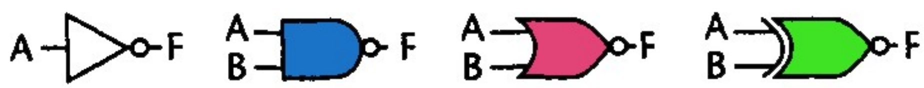
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

OR $F = A+B$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

XOR $F = A \oplus B$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



Inverter $F = \overline{A}$

A	F
0	1
1	0

NAND $F = \overline{AB}$

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

NOR $F = \overline{A+B}$

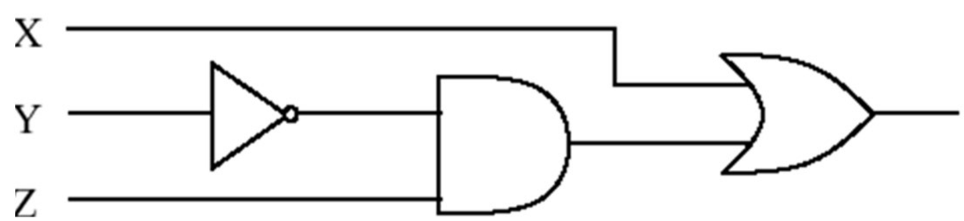
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

XNOR $F = \overline{A \oplus B}$

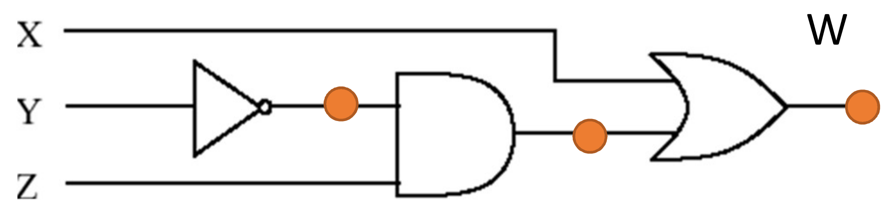
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Combinational Logic

- X, Y, Z are 3 digital input signals
- We can use a "Truth table" to find the output
- Because there are n=3 inputs: X, Y, Z
- So there are 2^n=8 rows in the truth table
  - Fill in Z: 0,1,0,1,0,1,0,1
  - Fill in Y: 0,0,1,1,0,0,1,1
  - Fill in X: 0,0,0,0,1,1,1,1



Truth table

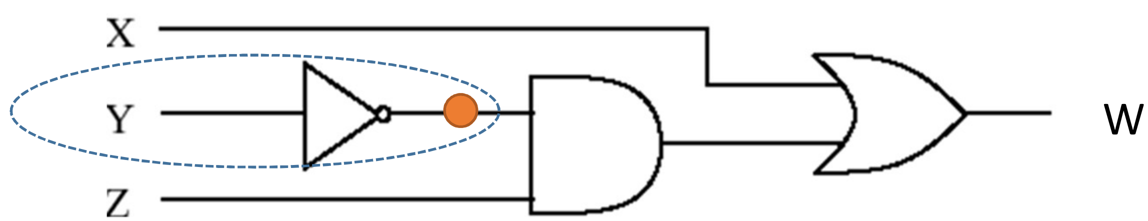


- We want to find: W = X OR (NOT(Y) AND Z)
- 3 inputs ➔ 2^3=8 rows
- Step 1: fill in different combinations of inputs

X	Y	Z			W=X OR (NOT (Y) AND Z)
0	0	0			?
0	0	1			?
0	1	0			?
0	1	1			?
1	0	0			
1	0	1			
1	1	0			
1	1	1			

We can solve it step by step

- Step2



X	Y	Z	NOT(Y)
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

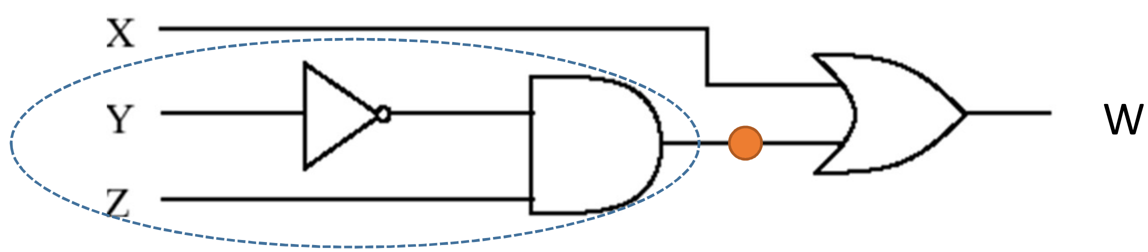
Produce NOT (Y) from Y first.
X,Z are not used in this step.

output

input

We can solve it step by step

- Step 3



X	Y	Z	NOT(Y)	Z AND (NOT(Y))
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

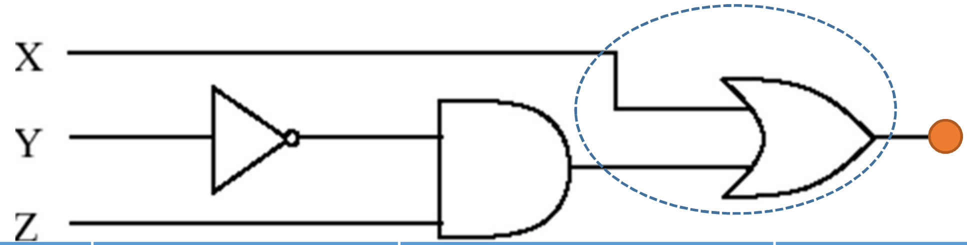
input input output

Then, produce [Z AND (NOT (Y))].

X , Y are not used directly in this step.

We can solve it step by step

- Step 4

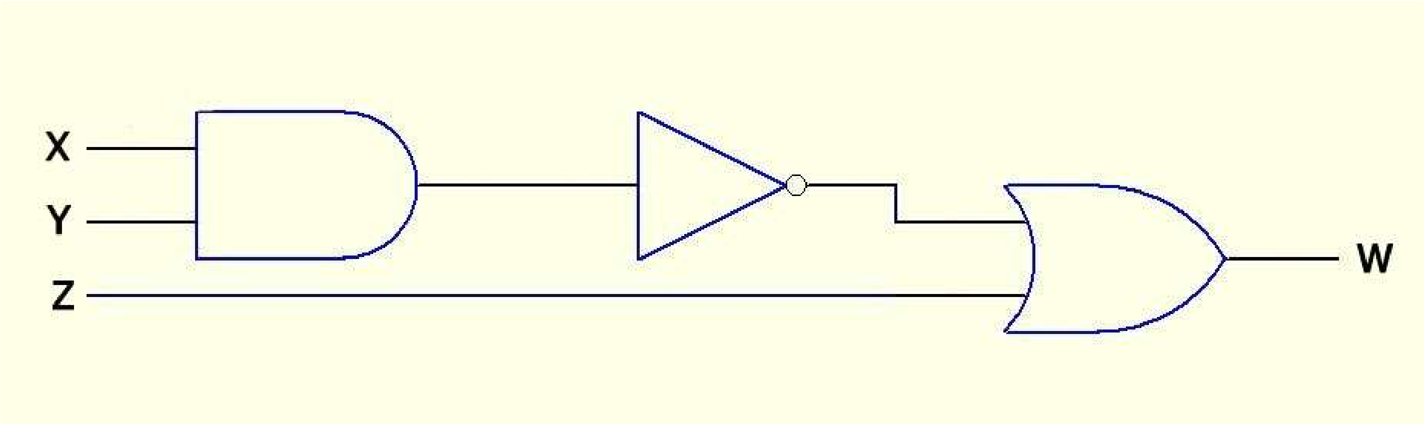


W=X OR (Z AND (NOT(Y)))

X	Y	Z	NOT(Y)	Z AND (NOT(Y)	W=X OR (Z AND (NOT(Y)))
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1 input	1	1	0	0 input	1 output

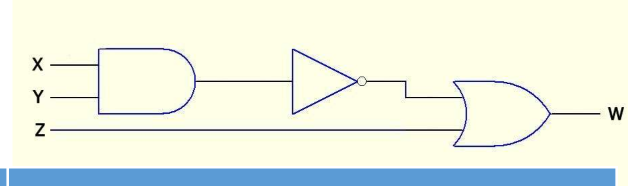
Exercise 1.1

- Use a truth table to find the output of
- NOT(X AND Y) OR Z



Exercise1.1: NOT(X AND Y) OR Z

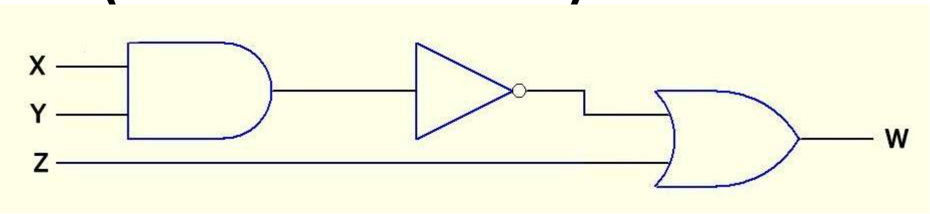
- Fill the blanks in X,Y, Z columns



X	Y	Z	X AND Y	NOT (X AND Y)	W=(NOT (Z AND Y)) OR Z
0		0			
		1			
		0			
		1			
		0			
		1			
		0			
		1			

Exercise1.1: NOT(X AND Y) OR Z

- Fill the blanks
- The answer is in the appendix



X	Y	Z	X AND Y	NOT (X AND Y)	W=(NOT (X AND Y)) OR Z
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

4. Software Implementation

Software Implementation

- Document about the use of Arduino
  - https://www.arduino.cc/en/Guide/HomePage
- Edit program
- Compile
- Download to the Arduino board of the robot
- Run the program

Example of Using Arduino

- How to Blink an LED with an Arduino Nano
- https://www.youtube.com/watch?v=dRAAIrpI1hg

- In this example, Pin 12 of Arduino Nano is used as digital output

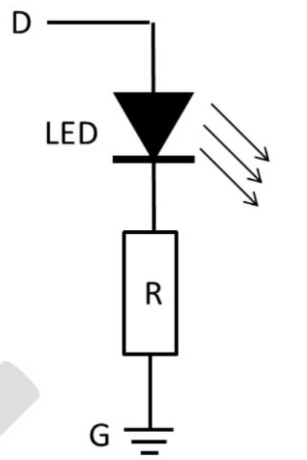


Figure-4.8: The LED circuit on the keyboard

How to Use "if-then-else"

- Suppose we want to express the following logic:
  - If A = 0, and B = 1,
  - Then C = 0,
  - Otherwise C = 1.

- Basic structure of if-then-else statement

```
if ( A==0 && B==1)
   C=0;
else
   C=1;
```

digitalRead and digitalWrite

- Suppose A0 and A1 are digital inputs, and A2 is a digital output of Arduino

```
if (digitalRead(A0)==0 && digitalRead(A1)==1)
      digitalWrite(A2,0);
   else
      digitalWrite(A2,1);
```

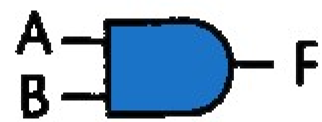
Simple Logic Implementation

- Setup the I/O pins in the "setup" function:

```
void setup()
{
   pinMode(A0, INPUT);    // refers to A
   pinMode(A1, INPUT);    // refers to B
   pinMode(A2, OUTPUT);   // refers to F
}
```

- Implement the logic in the "loop" function:

```
void loop()
{
   if (digitalRead(A0)==1 && digitalRead(A1)==1)
      digitalWrite(A2,1);
   else
      digitalWrite(A2,0);
}
```



AND

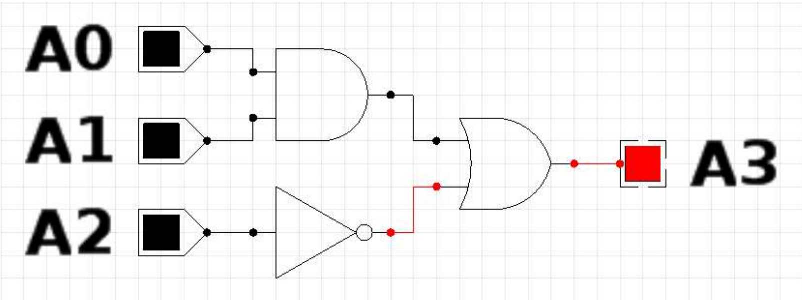
F = AB

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Combinational Logic Implementation

- Setup the I/O pins in the “setup” function:

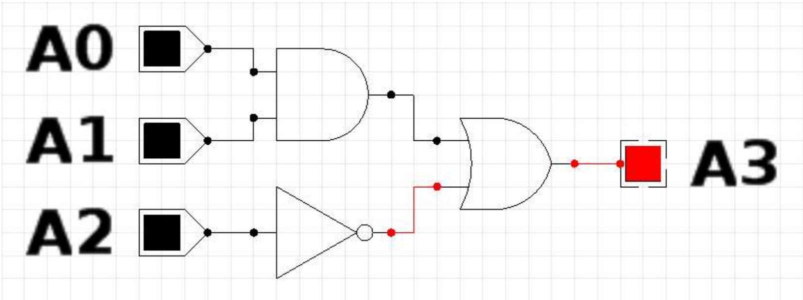
```
void setup()
{
    pinMode(A0, INPUT);
    pinMode(A1, INPUT);
    pinMode(A2, INPUT);
    pinMode(A3, OUTPUT);
}
```



Combinational Logic Implementation

- Implement the logic in the “loop” function:

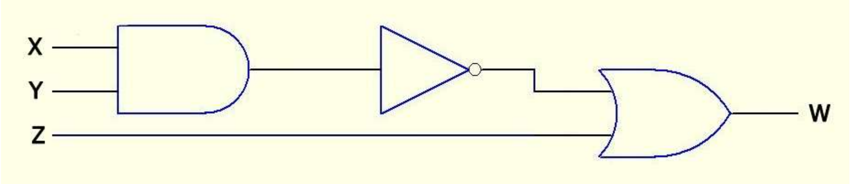
```
void loop()
{
   if ( (digitalRead(A0)==1 && digitalRead(A1)==1) ||
         !(digitalRead(A2)==1 )

      digitalWrite(A3,1);
   else
      digitalWrite(A3,0);
}
```



END

Appendix 1

- ANSWER1.1: W=(NOT(X AND Y)) OR Z
- Fill the blanks



X	Y	Z	X AND Y	NOT (X AND Y)	W=(NOT (X AND Y)) OR Z
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	0	1