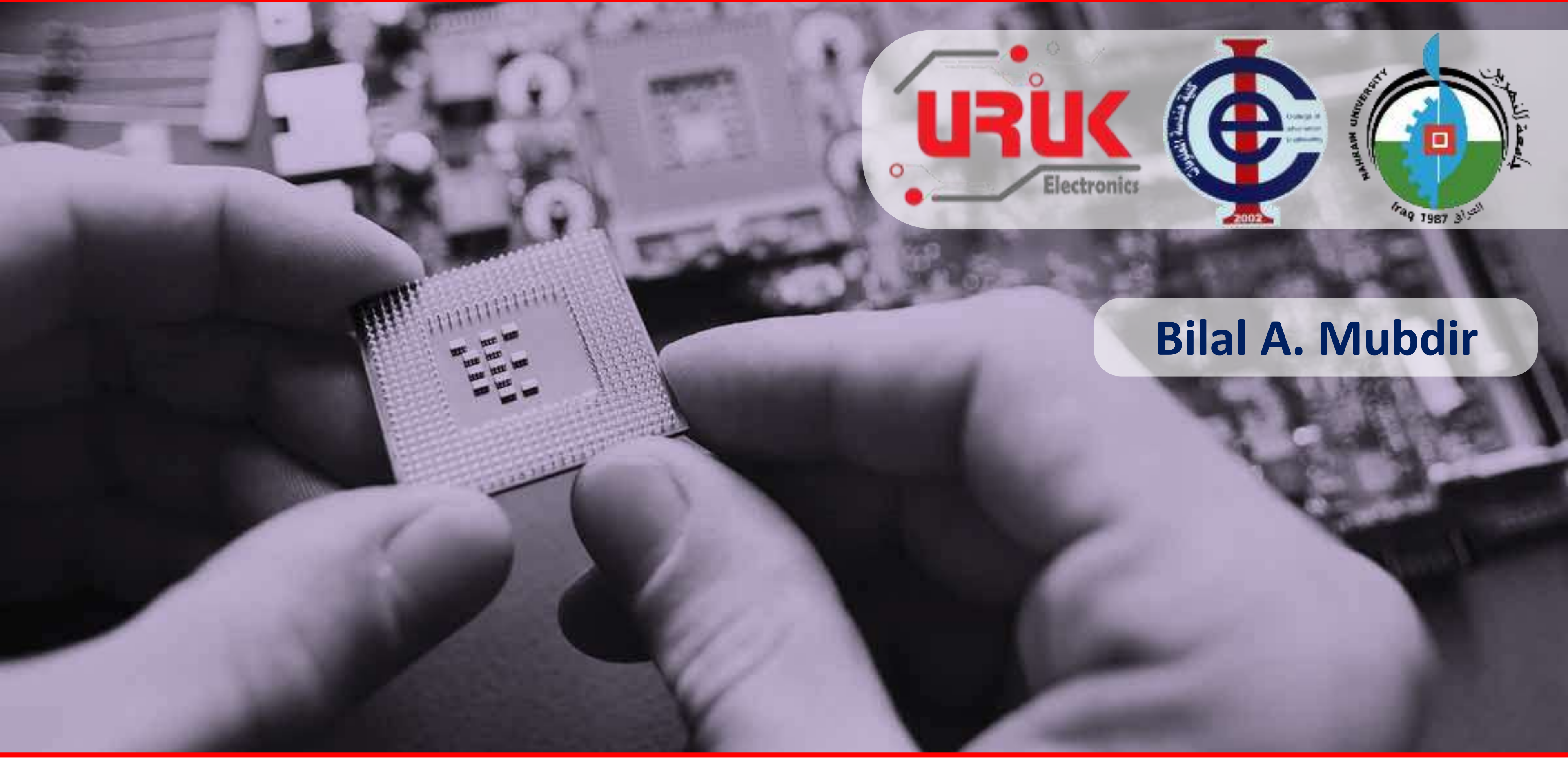
Workshop on **Real Time Control & Embedded System**

Microcontroller based Real Time Embedded System via MATLAB



Bilal A. Mubdir

Training & Development Department
training@uruktech.com

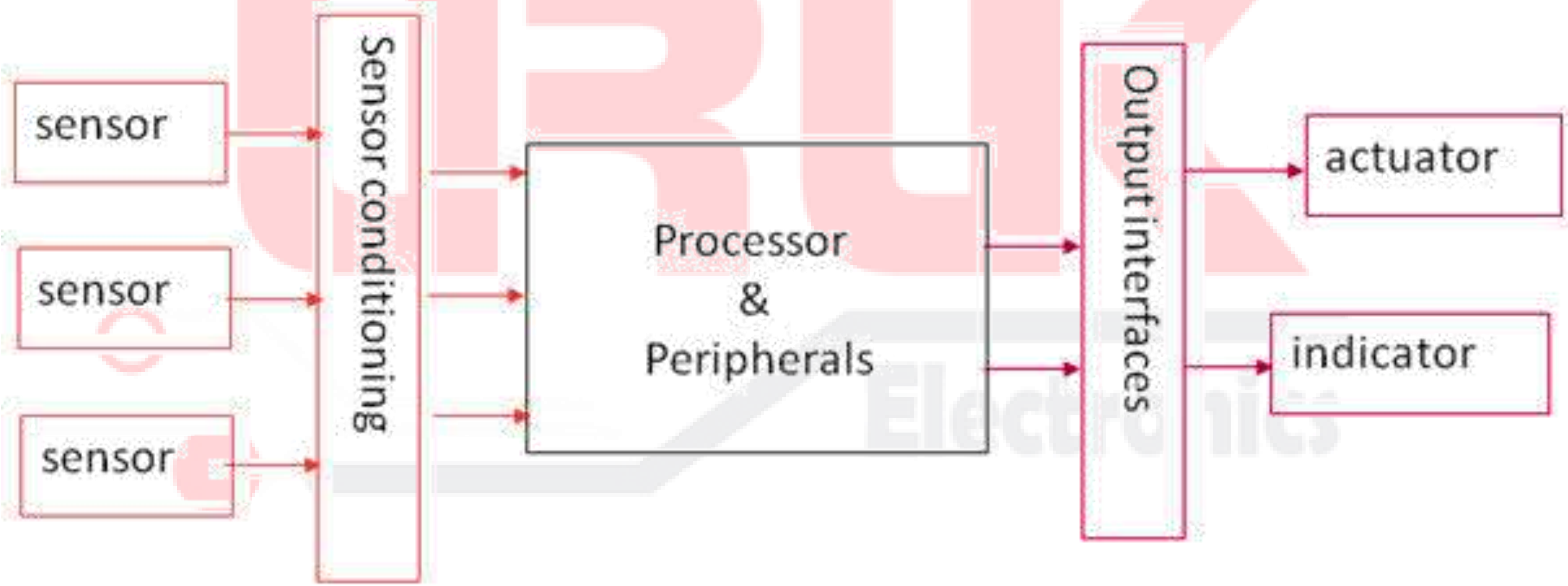
www.uruktech.com

Lecture Contents

- ❑ What is Embedded System?
- ❑ What is Real Time System?
- ❑ Embedded System/Case Study
- ❑ DC Motor
- ❑ System Objectives
- ❑ Controller
- ❑ System Overview
- ❑ Driving DC Motor
- ❑ Tachometer
- ❑ Arduino Board/DAQ
- ❑ Interrupt in Arduino
- ❑ Computer/MATLAB
- ❑ Conclusions

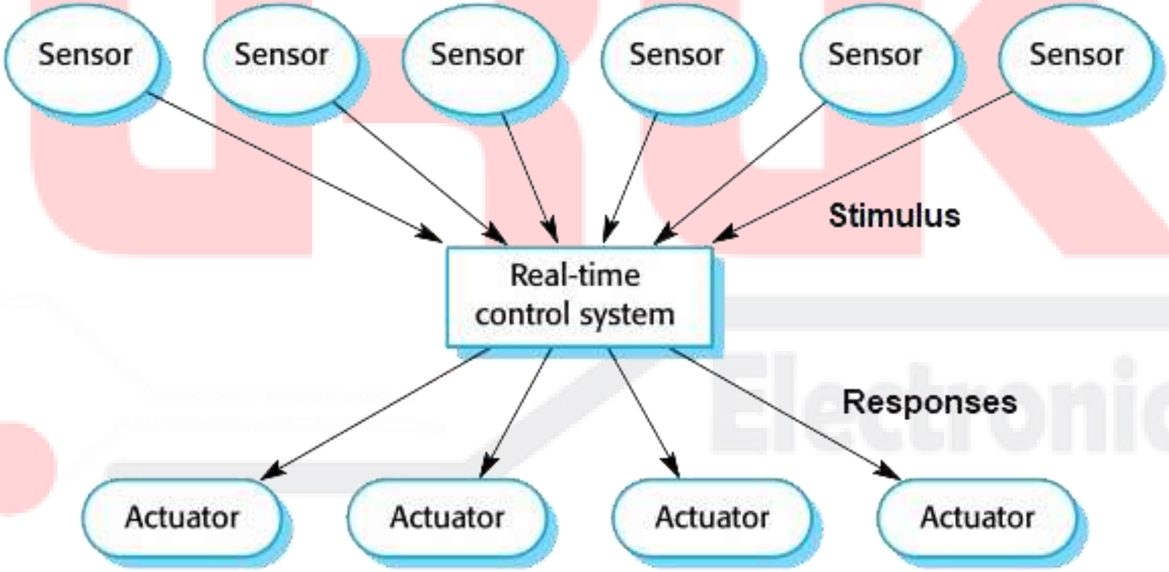
Embedded System

An **embedded system** is a **computer system** with a **dedicated function** within a larger **mechanical or electrical system**, often with real-time computing constraints.



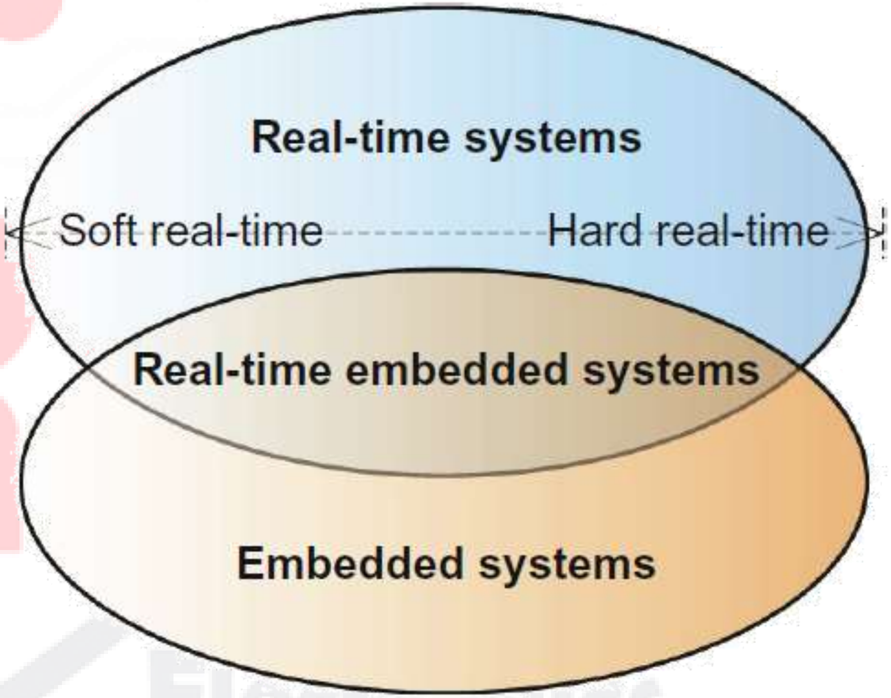
Real Time System

A real-time system is a type of hardware/software that operates with a **time constraint**, on which "controls an environment by **receiving data**, **processing them**, and **returning the results sufficiently quickly** to affect the environment **at that time**".



Real Time System

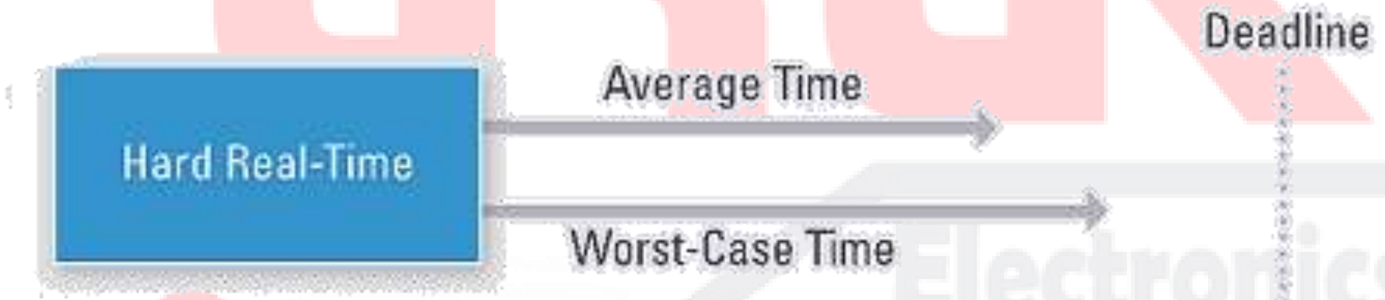
Classification



Real Time System

Hard Real Time

A timing constraint is **hard** if the consequence of a missed deadline is fatal. A late response (completion of the requested task) is useless, and sometimes totally unacceptable.



Real Time System

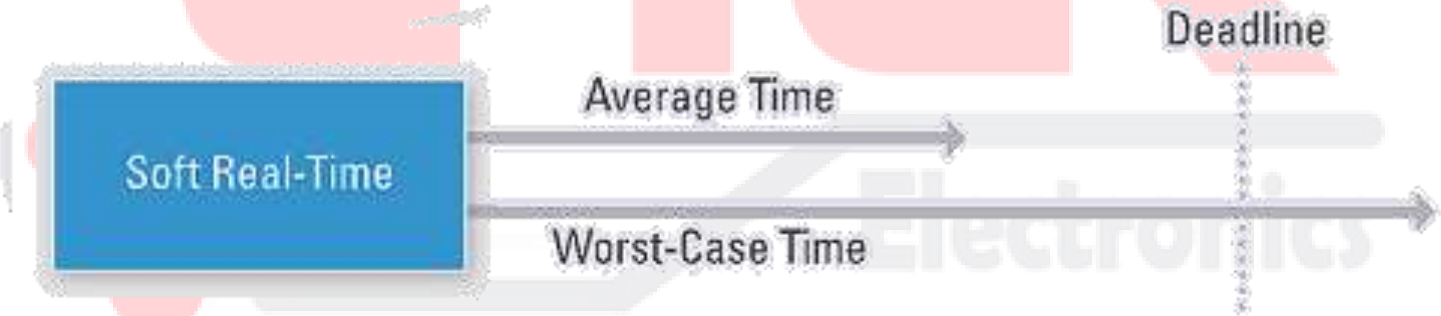
Hard Real Time

Example System	Example Timing Constraint	Consequence of Missed Deadlines
Antilock braking system	The antilock braking system should apply/release braking pressure 15 times per second a wheel that locks up should stop spinning in less than 1s	Loss of human lives
Antimissile system	It never needs more that 30 s to intercept a missile after it reenters the atmosphere (in the terminal phase of its trajectory)	Loss of human lives, huge financial loss
Cardiac pacemaker	The pacemaker waits for a ventricular beat after the detection of an atrial beat. The lower bound of the waiting time is 0.1 s, and the upper bound of the waiting time is 0.2 s	Loss of human life
FTSE 100 Index	It is calculated in real time and published every 15 s	Financial catastrophe

Real Time System

Soft Real Time

A timing constraint is **soft** if the consequence of a missed deadline is undesirable but tolerable. A late response is still useful as long as it is within some acceptable range (say, it occurs occasionally with some acceptably low probability).

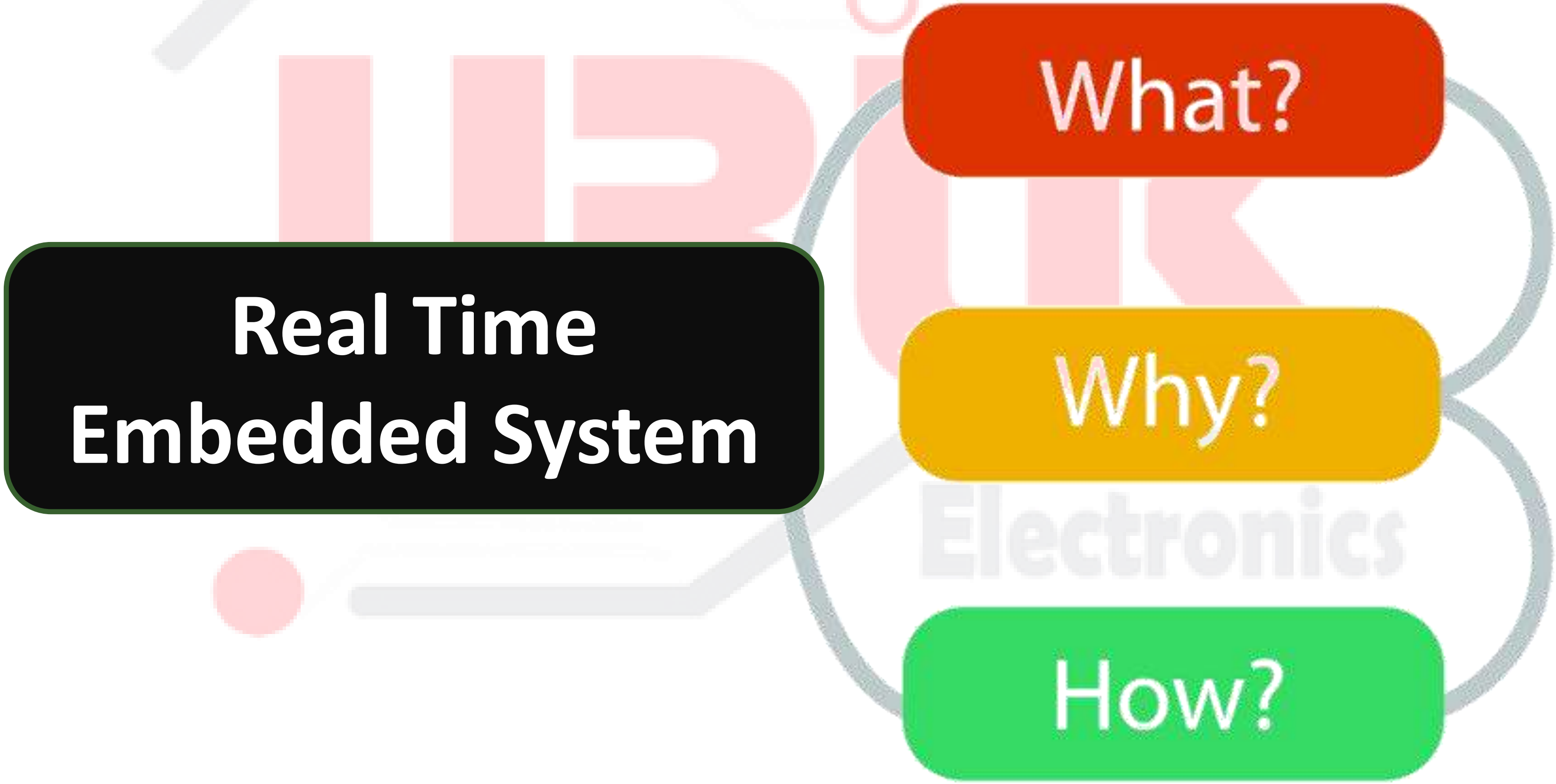


Real Time System

Soft Real Time

Example System	Example Timing Constraint	Consequence of Missed Deadlines
Digital camera	Shutter speed, shown in seconds or fractions of a second, is a measurement of the time the shutter is open. When the shutter speed is set to 0.5 s, the shutter open time should be (0.5 ± 0.125)s 99.9% of the time	Unsatisfied users may switch to other models
Global positioning system	Upon identifying a waypoint, it can remind the driver at a latency of 1.5 s	The driver misses the waypoint
Robot-soccer player	Once it has caught the ball, the robot needs to kick the ball within 2 s, with the probability of breaking this deadline being less than 10%	Its team may lose the game
Wireless router	The average number of late/lost frames is less than 2/min	The user has bad Web surfing experience

Embedded System/Case Study



Real Time Embedded System

What?

Why?

How?

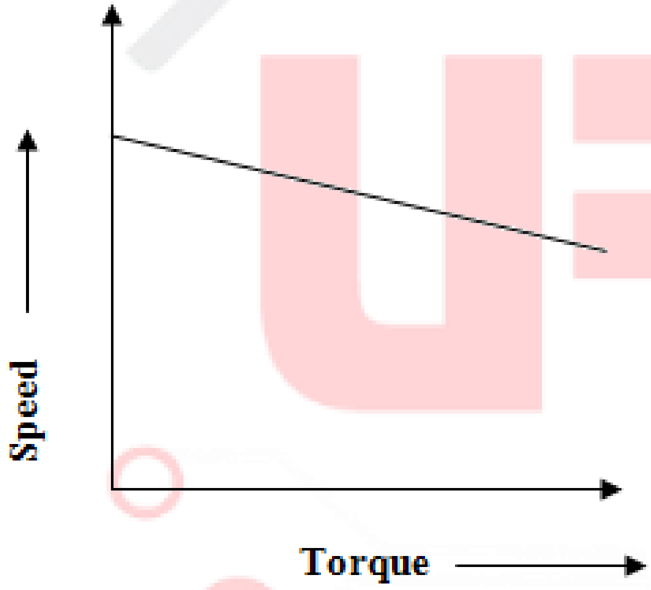
Embedded System/Case Study

What?

Speed of a DC Motor Problem!

Embedded System/Case Study

Why?



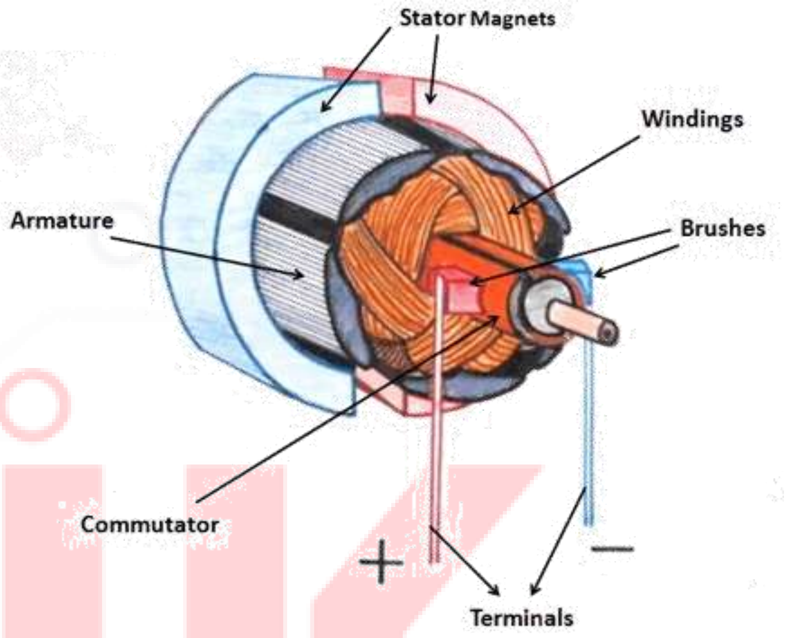
In most applications, it is required to **change motor’s speed rate** to a **pre-defined value** for performing a specific work process and **keeping the speed constant** at that rate even if there is any external disturbances.

Embedded System/Case Study

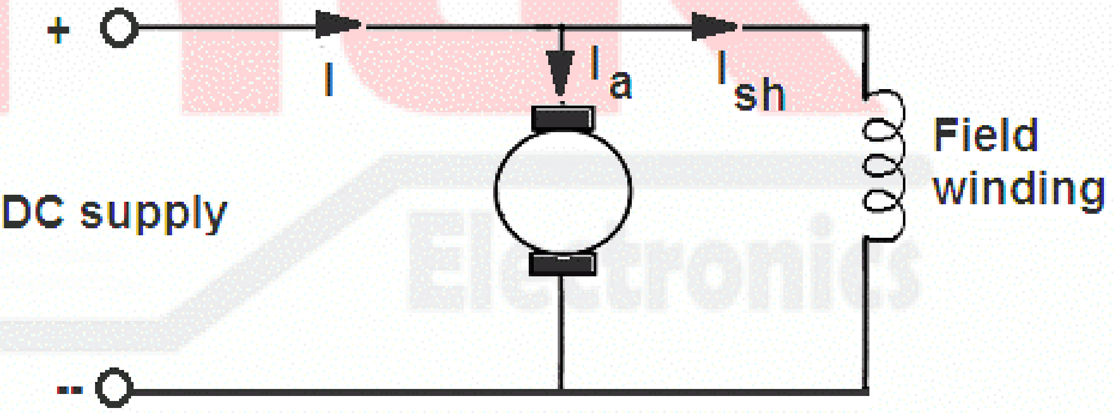
How?

Designing and implementing an “**Embedded System for Real Time Speed Control of DC Motor**”.

DC Motor



$$E_b = \frac{\Phi Z N}{60} \bullet \frac{P}{A} \, Volts$$



Where,

$$N \propto V$$

DC Motor

Speed Control of DC Motor

❑ Flux control method

❑ Armature and Rheostatic control method

❑ Voltage control method ← **PWM**

PARAMETERS	FLUX CONTROL METHOD	ARMATURE CONTROL METHOD	PULSE WIDTH MODULATION
POWER EFFICENCY	Good	High power loss	The power efficiency is high
SPEED CONTROL BEHAVIOUR	Only speed above base speed can be controlled	Speed control is possible	A precise speed control is achieved
CONTROL CIRCUIT	Very large	Very large	Since it uses electronic circuit, it is compact

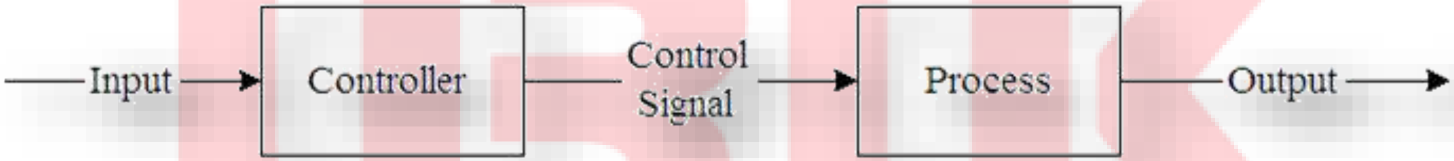
System Objectives

- ❑ **Controlling the Speed** of a DC motor at any pre-defined rate.
- ❑ Operating the DC Motor in **two modes of control**.
- ❑ Keeping a DC motor driven at a **constant speed** in case of any disturbance (Closed Loop Control).
- ❑ Using **Personal Computer** (Main Microprocessor) to govern the system.
- ❑ Using **Arduino Board** (Atmel Microcontroller) as Data Acquisition System.
- ❑ Design a **GUI** to control the **desired speed** and **controller parameters**.

Controller

Modes

An **Open-loop System** referred to as non-feedback system, is a type of continuous control system in which the output has no influence or effect on the control action of the input signal.

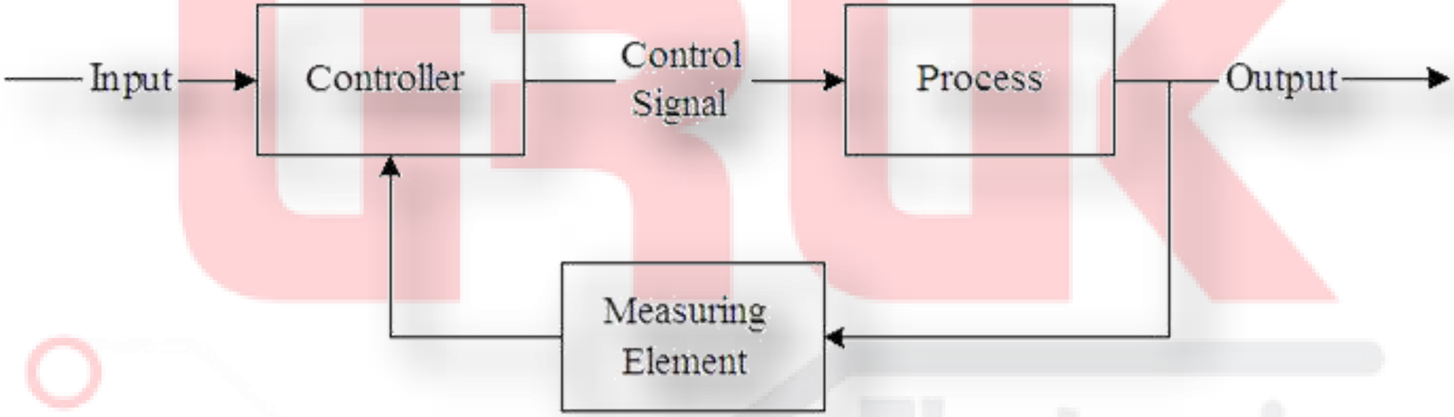


- Open-loop system has **no knowledge of the output condition** so **cannot self-correct** any errors it could make when the preset value drifts.
- Open-loop systems are poorly equipped to handle **disturbances or changes** in the conditions which may reduce its ability to complete the desired task

Controller

Modes

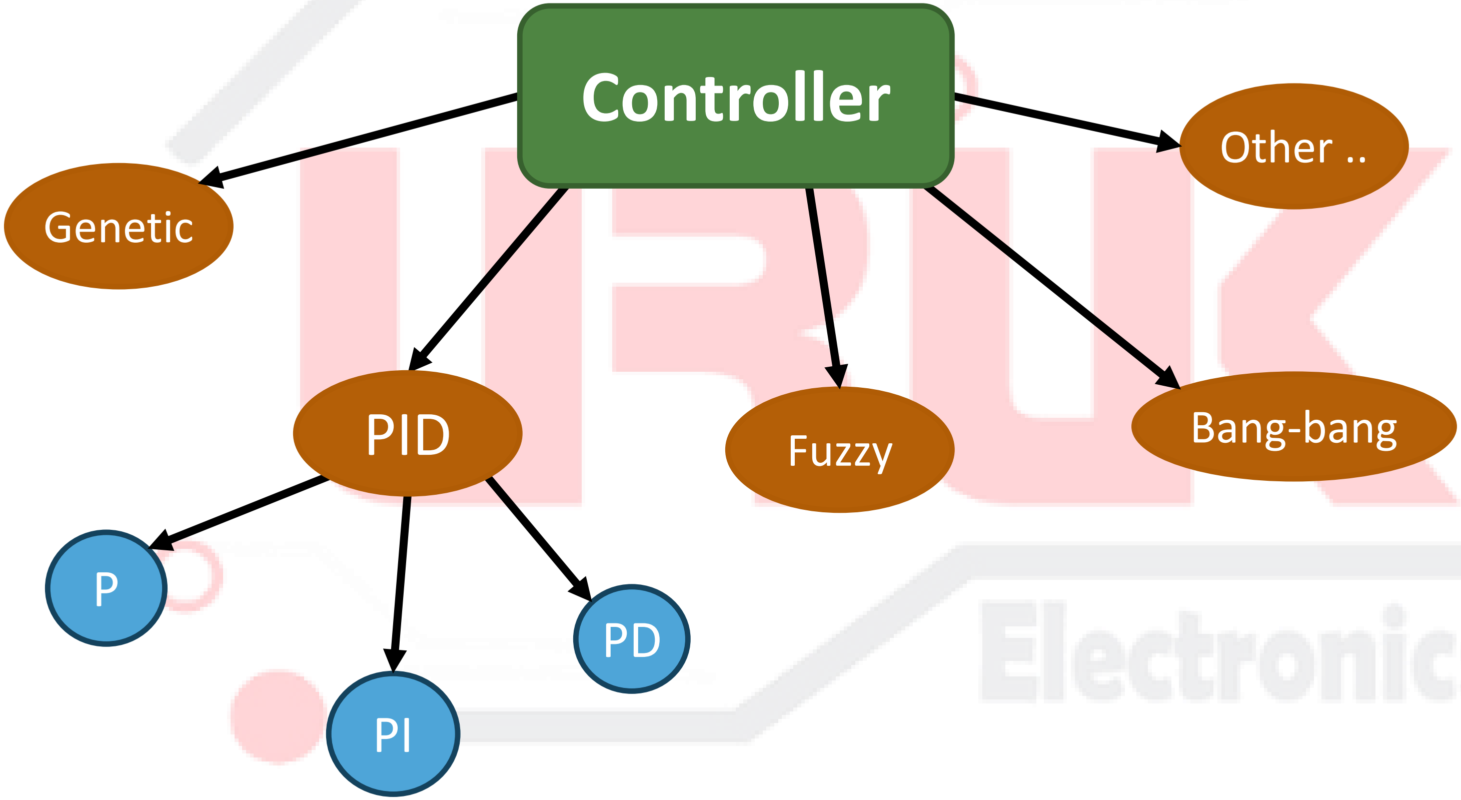
A **Closed-loop Control System**, is a feedback control system which uses the concept of an open loop system as its forward path but has one or more paths between its output and its input.



The reference to “feedback”, simply means that some portion of the output is returned “back” to the input to form part of the systems excitation.

Controller

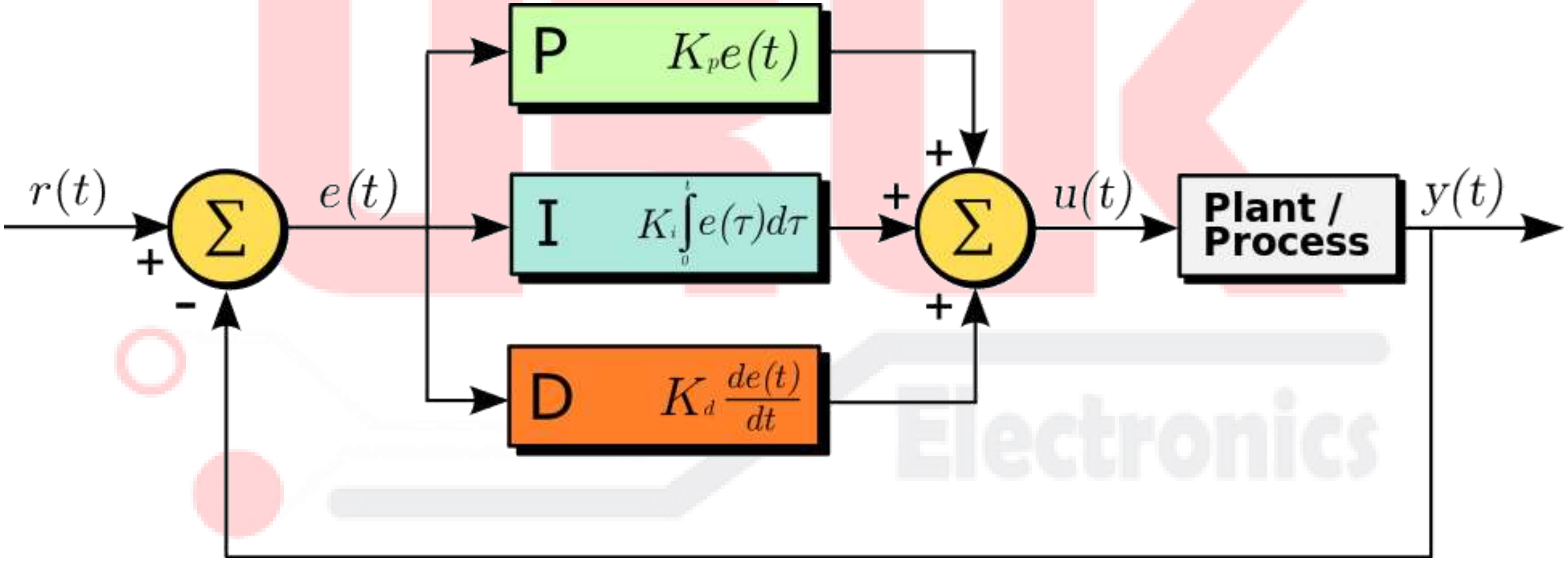
Types



Controller

PID

A combination of **proportional**, **integral** and **derivative** actions is more commonly referred as **PID** action and hence the name, PID controller.

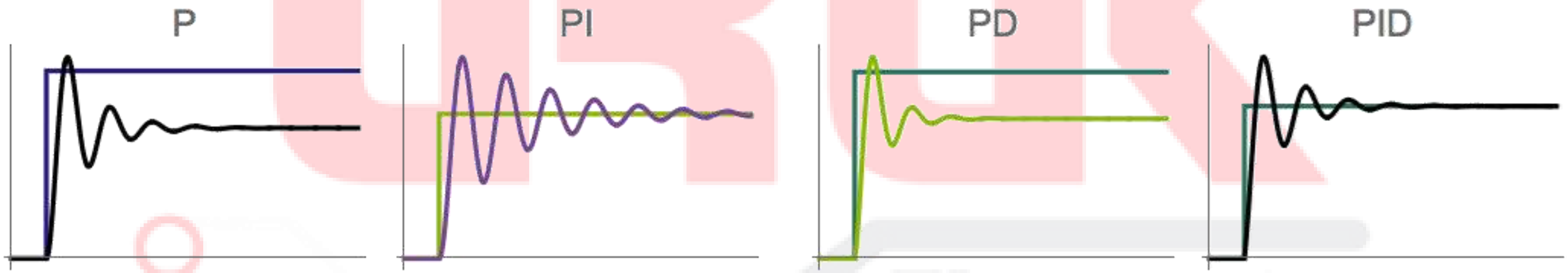


Controller

PID

PID controllers have three control modes:

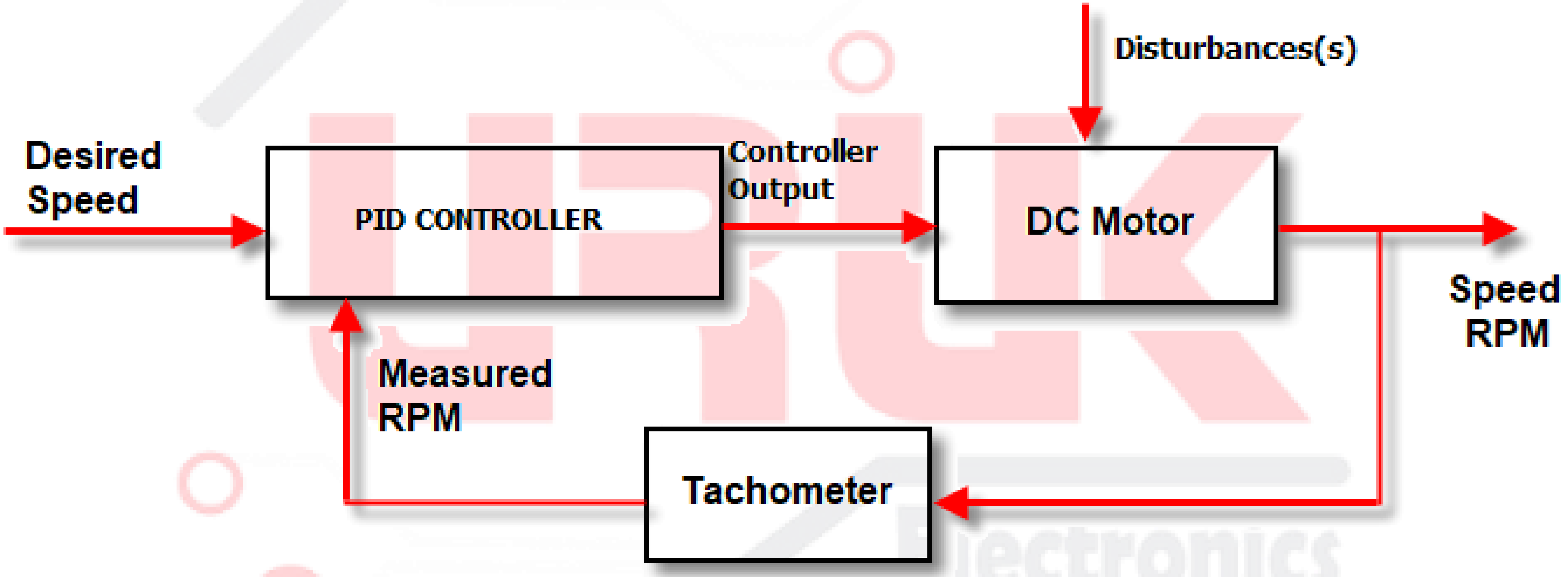
- **Proportional Control**
- **Integral Control**
- **Derivative Control**



Each of the three modes reacts differently to the error. The amount of **response produced** by each control mode is **adjustable** by changing the controller's **tuning settings**.

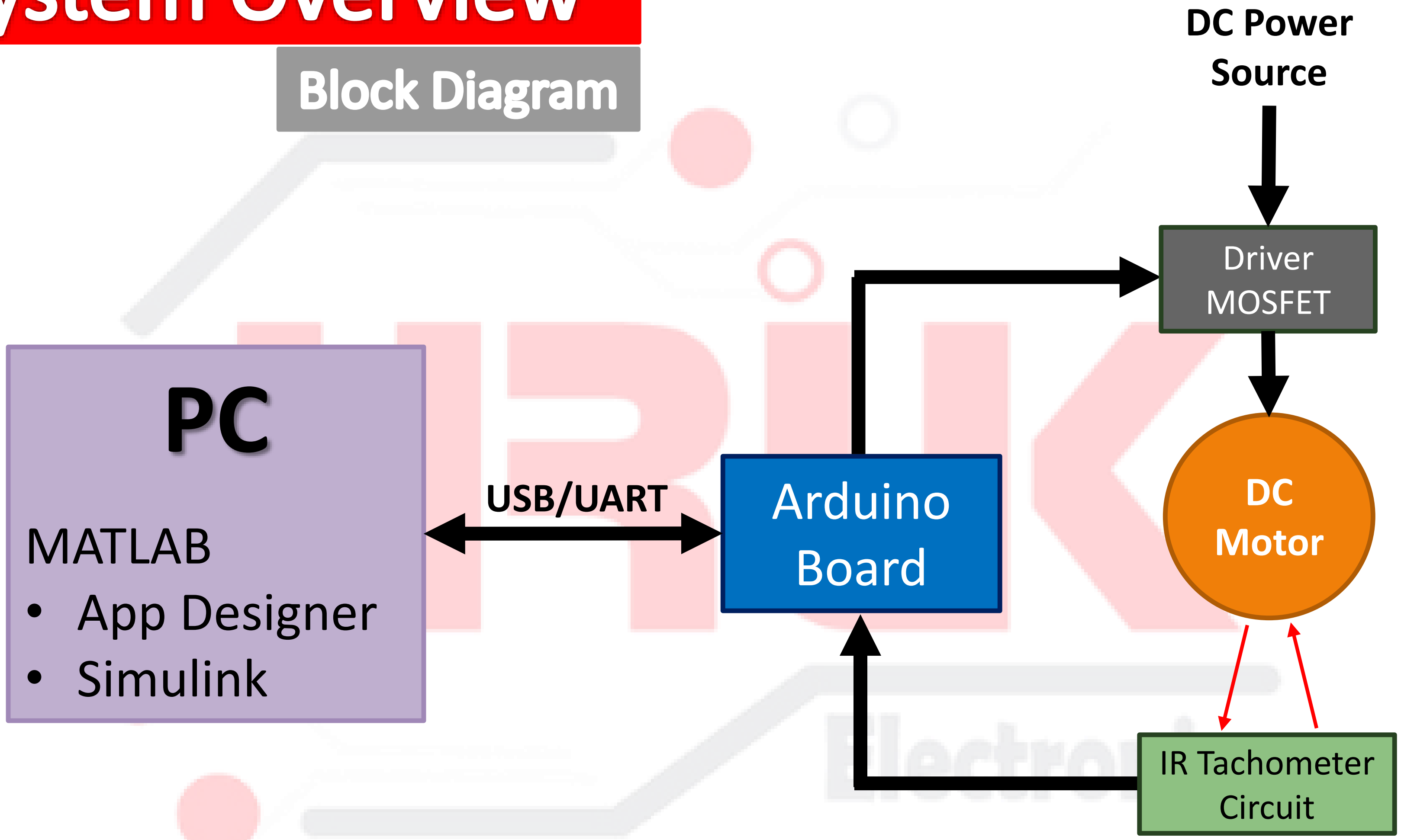
Controller

PID



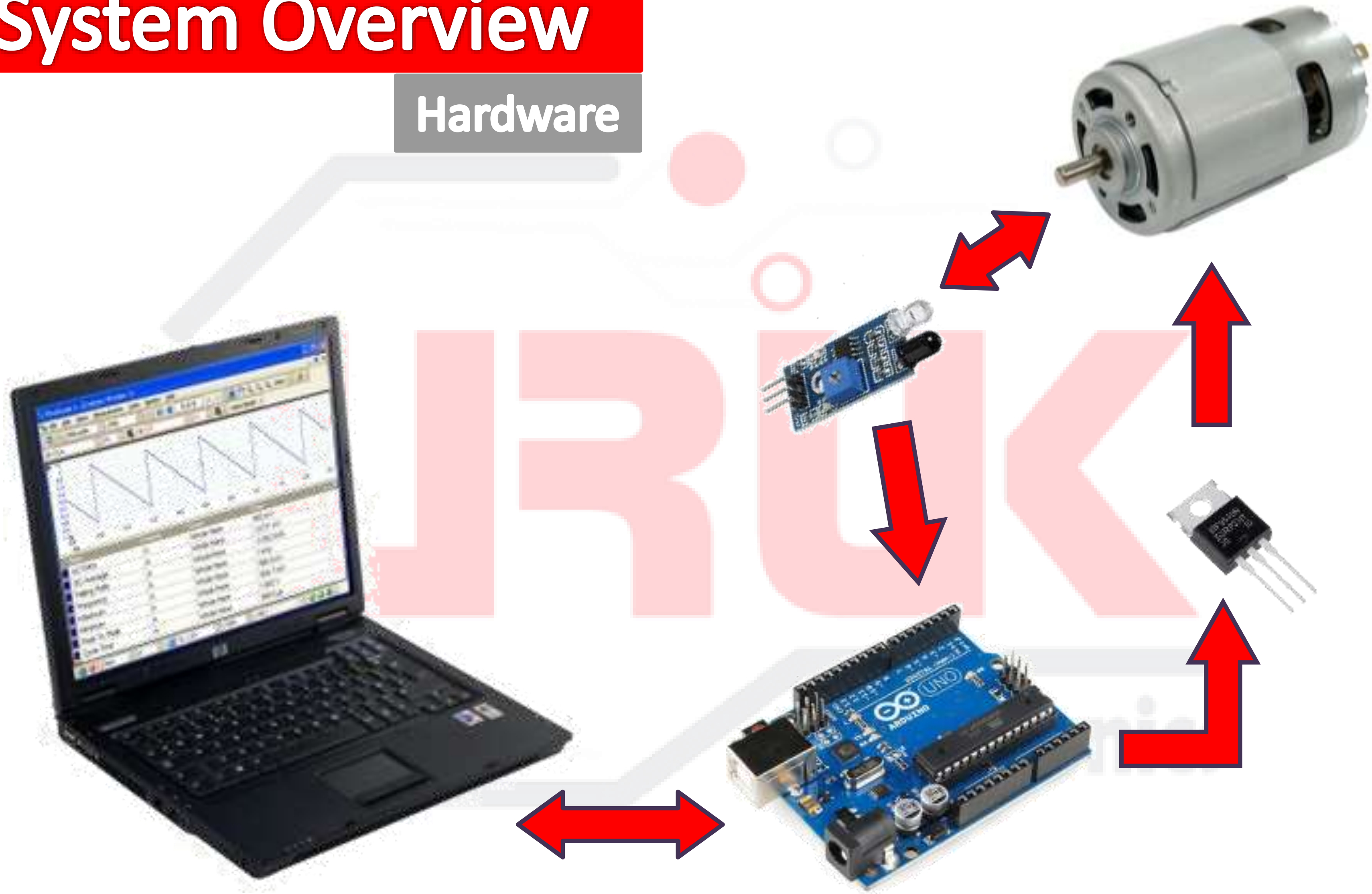
System Overview

Block Diagram



System Overview

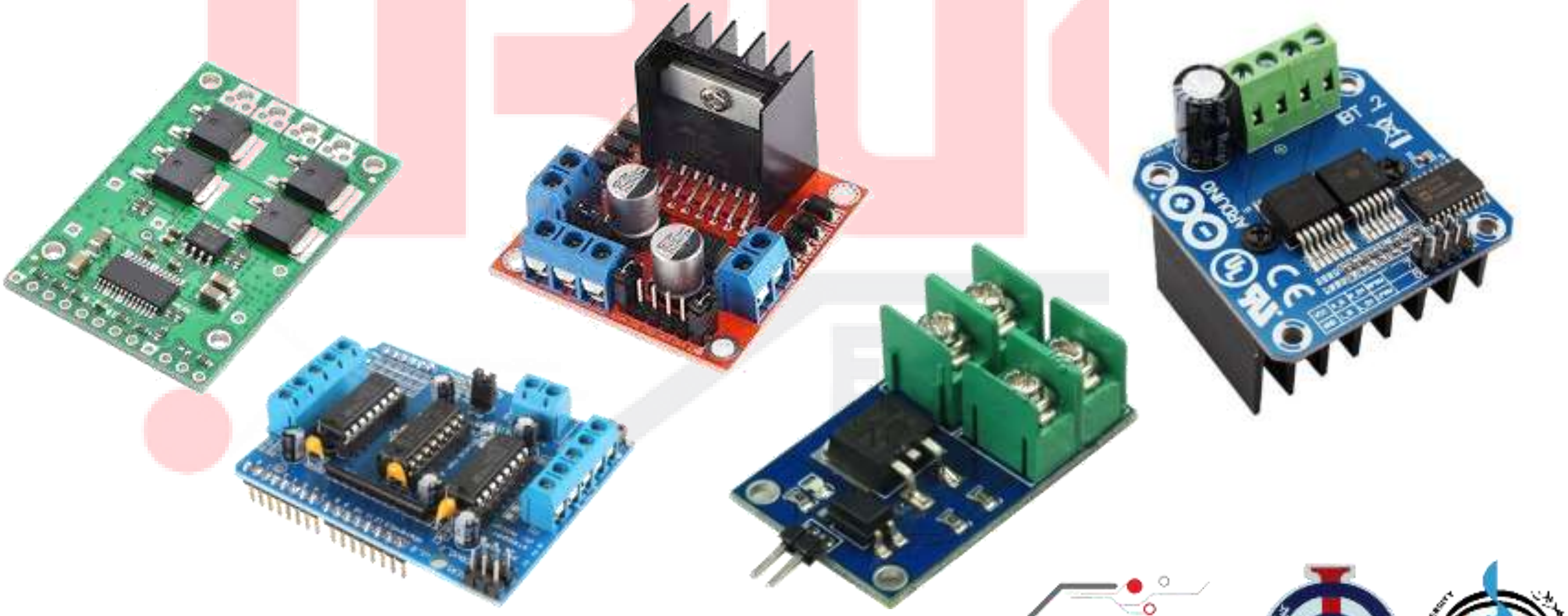
Hardware



Driving DC Motor

Driver Modules

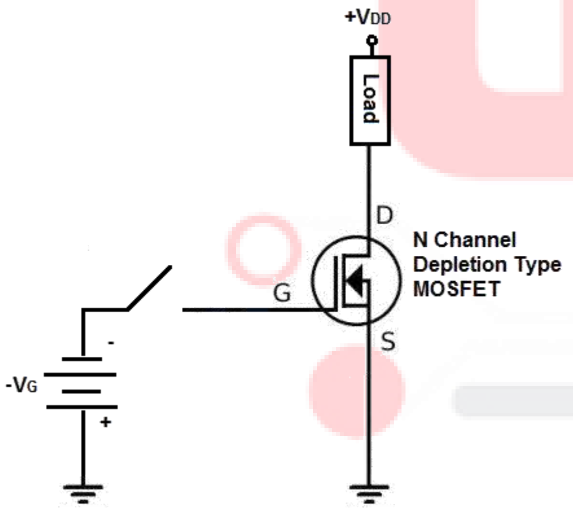
A motor driver is **current amplifier**; the function of motor drivers is to take a **low-current control signal** and then turn it into a **higher-current signal** that can drive a motor.

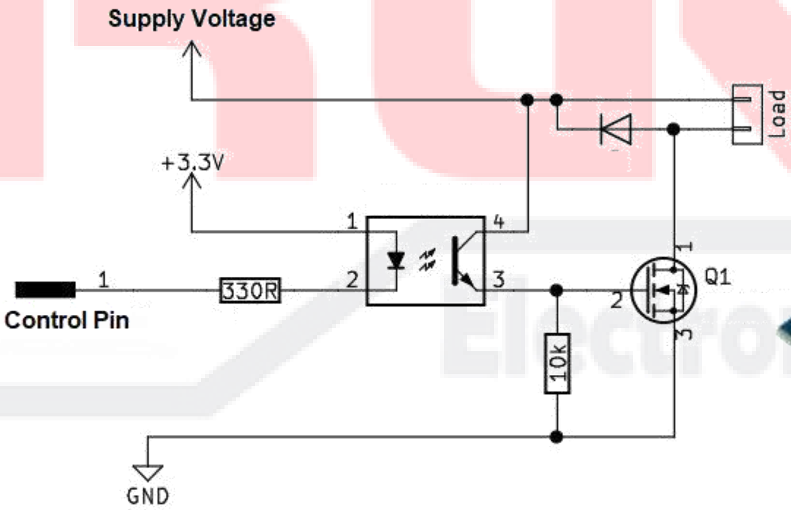


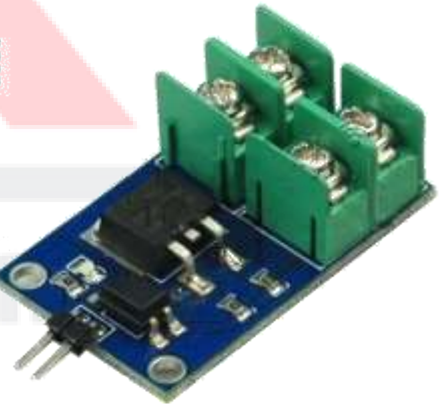
Driving DC Motor

MOSFET

Well, a MOSFET is like a **voltage-controlled switch**. To be more precise, an N-channel enhancement type MOSFET is like an infinite resistance when the gate-to-source voltage is zero, and turns into a very low resistance when the gate-to-source voltage is a few volts positive.





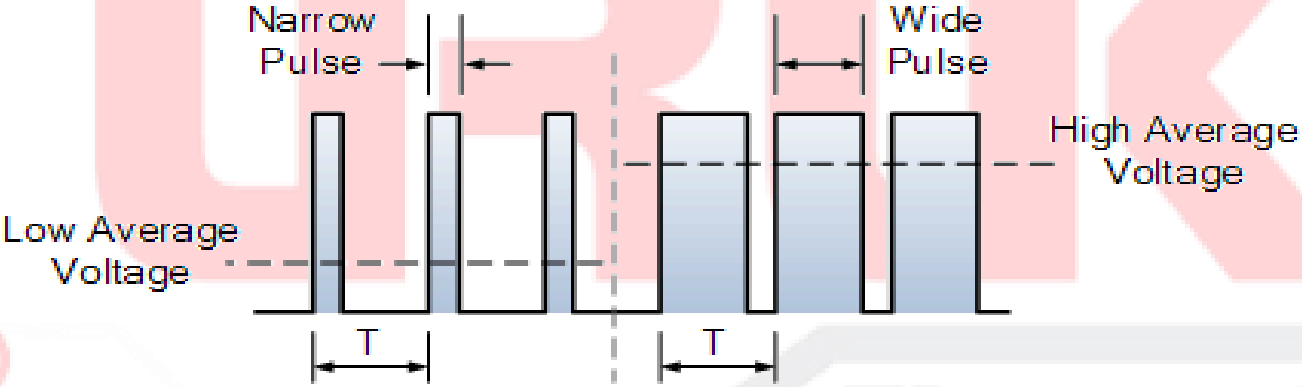


Driving DC Motor

PWM

- ❑ An effective method to **control the output voltage** with constant frequency.
- ❑ This is a modulation of pulses by varying the duty cycle.

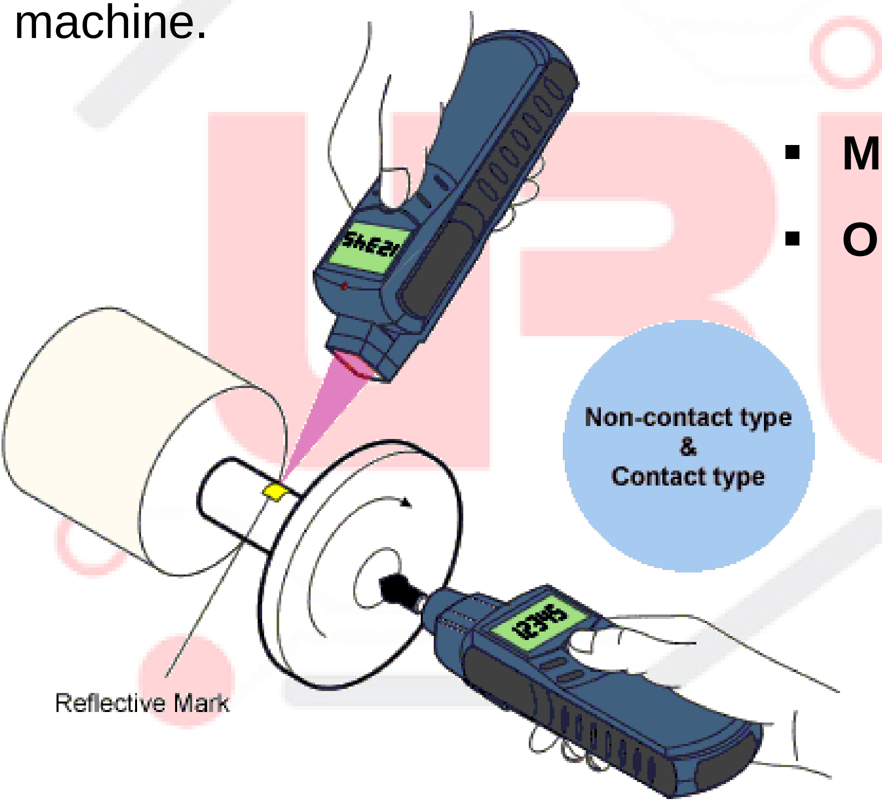
$$Duty\ Cycle\ (\alpha) = \frac{T_{ON}}{T_{ON}+T_{OFF}}$$



- ❑ The width of pulses (**T**) determines the amount of avg. voltage applied to the DC motor terminals.

Tachometer

A tachometer (revolution-counter, tach, rev-counter, RPM gauge) is an instrument **measuring the rotation speed** of a shaft or disk, as in a motor or other machine.



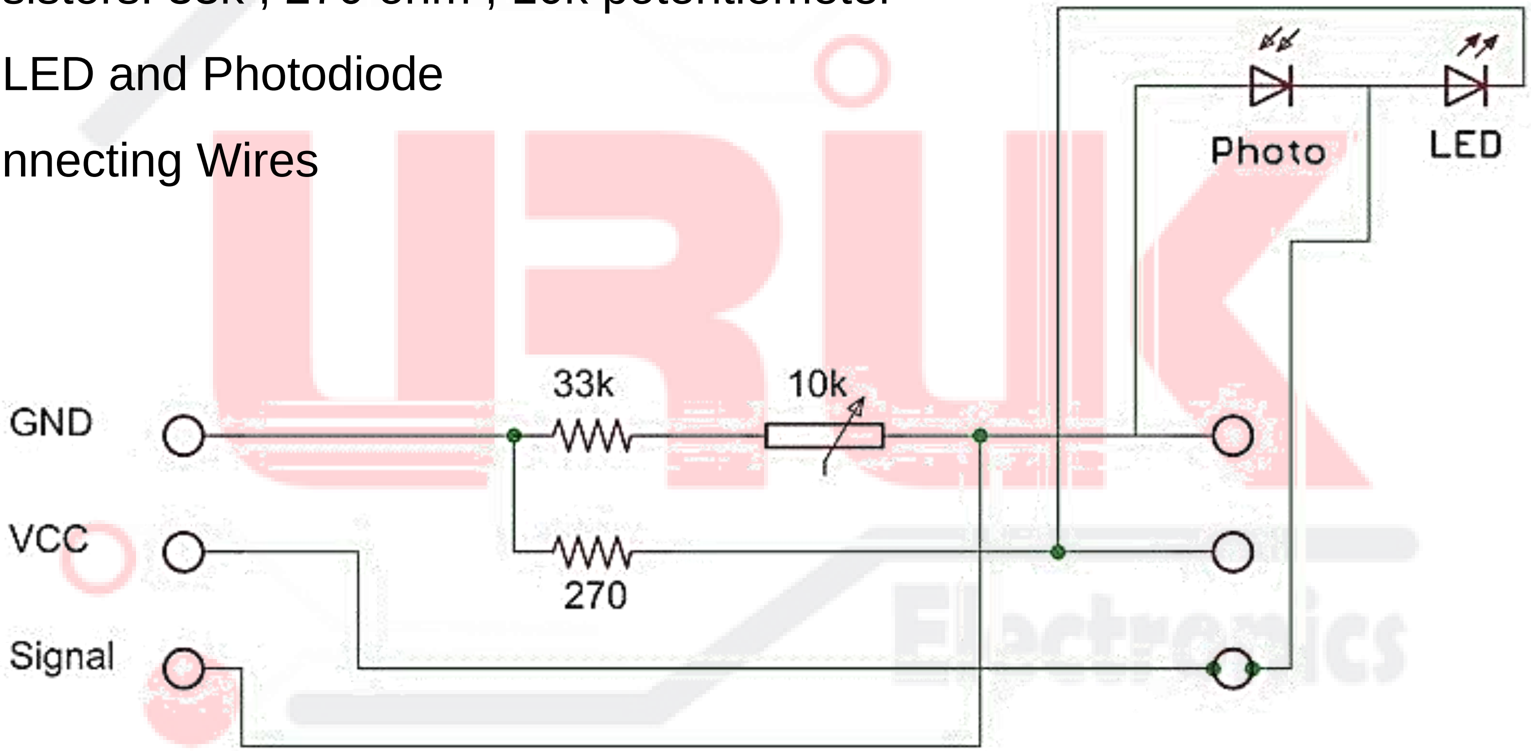
Non-contact type
&
Contact type

- **Mechanical Type**
- **Optical Type** (Laser Beam or IR)

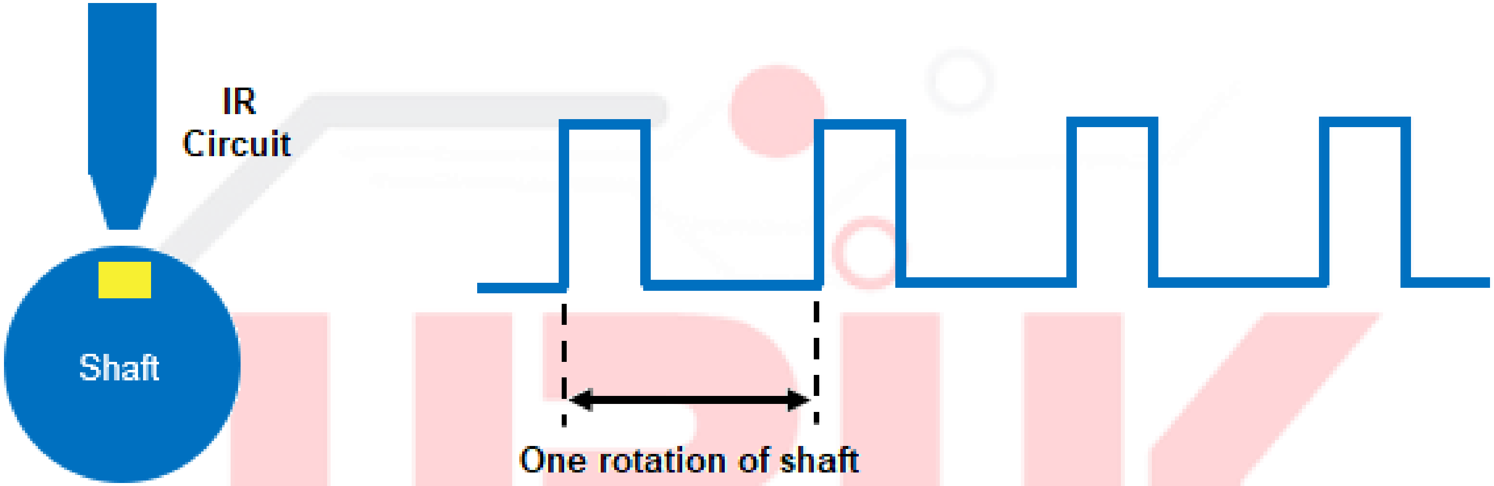
Tachometer

Simple Tachometer IR Sensor Circuit

- Resistors: 33k , 270 ohm , 10k potentiometer
- IR LED and Photodiode
- Connecting Wires



Tachometer



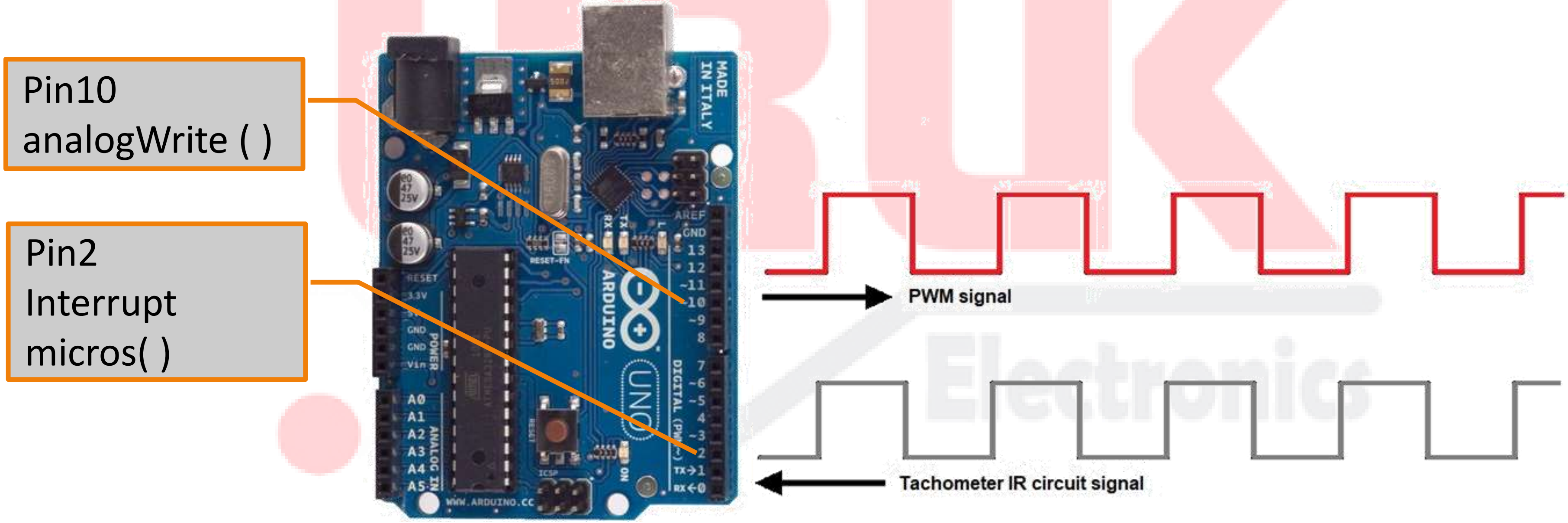
$$\textbf{Time for one rotation} = \text{Pulse Time (P\_time)} \qquad \textit{seconds}$$

$$\textbf{Motor Speed (RPM)} = \frac{\text{No. of Rotations}}{\text{One Minute}}$$

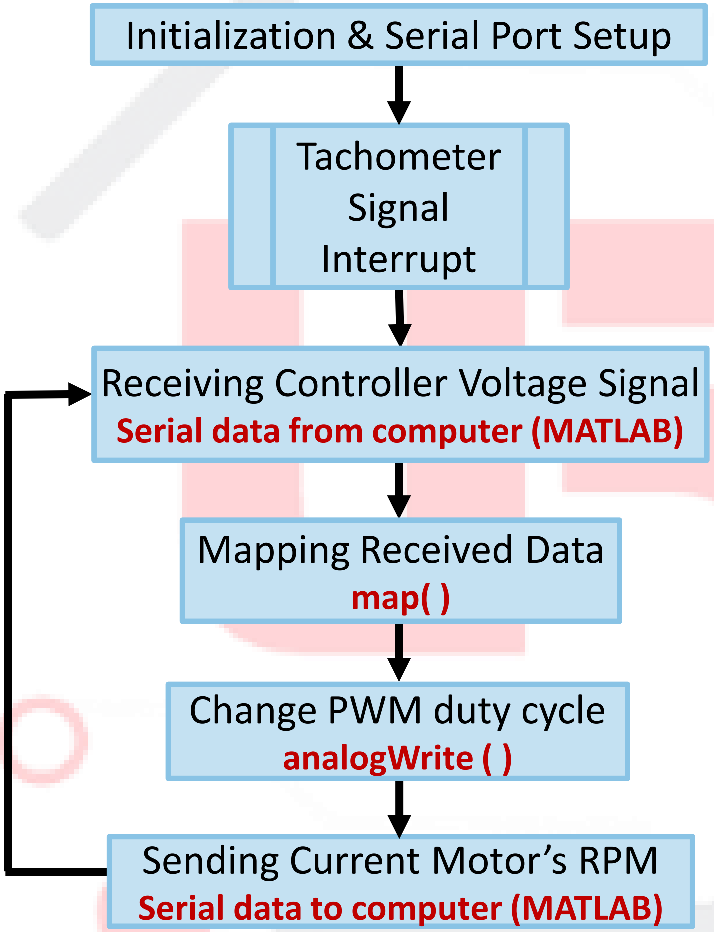
$$= \frac{60}{\text{Pulse Time}} \qquad \textit{revolution per minute}$$

Arduino Board/DAQ

The Arduino Uno board (Atmel Microcontroller) has been used as **Data Acquisition System (DAQ)** to **send/receive** date **to/from** the computer. Also, the tachometer's calculations and the PWM signal generation processed by the Arduino board.



Arduino Board/DAQ



Initialization & Serial Port Setup

Tachometer Signal Interrupt

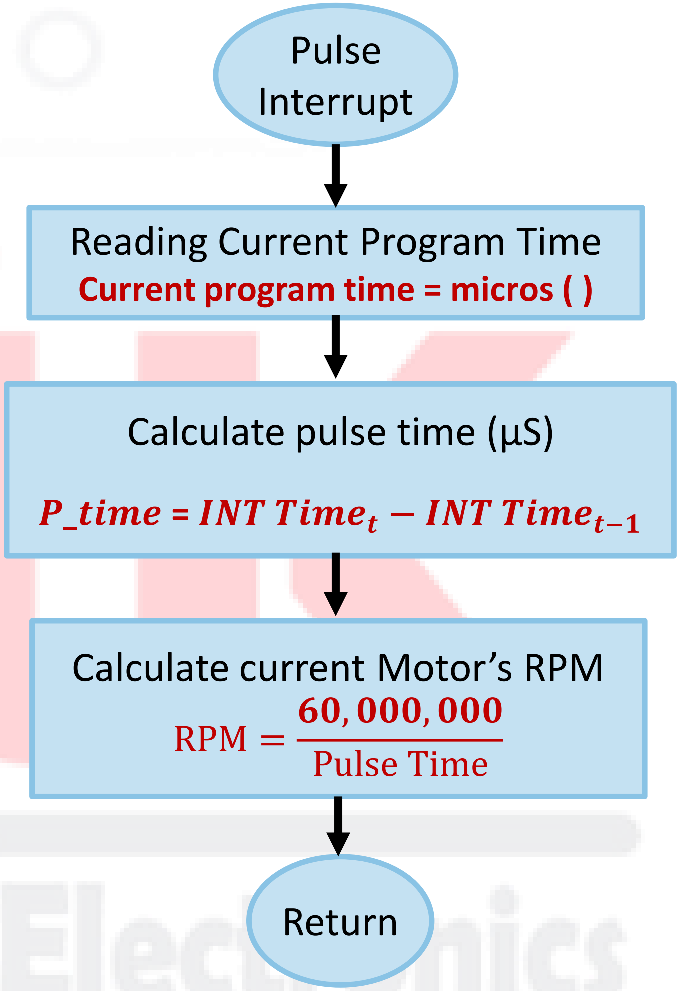
Receiving Controller Voltage Signal
Serial data from computer (MATLAB)

Mapping Received Data
map( )

Change PWM duty cycle
analogWrite ( )

Sending Current Motor’s RPM
Serial data to computer (MATLAB)

Ardunio Board’s Main Program



Pulse Interrupt

Reading Current Program Time
Current program time = micros ( )

Calculate pulse time (µS)

$$P\_time = INT\ Time_t - INT\ Time_{t-1}$$

Calculate current Motor’s RPM

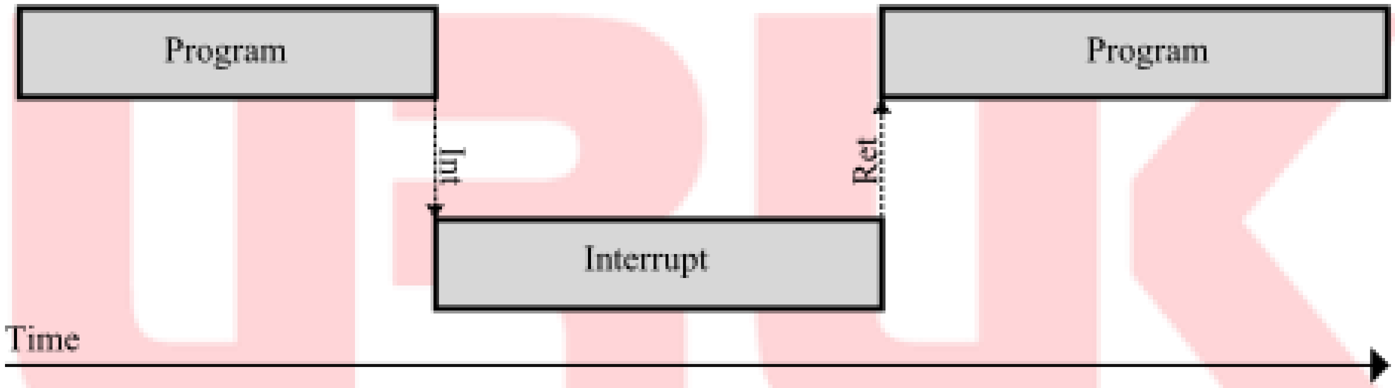
$$\mathrm{RPM} = \frac{60,000,000}{\mathrm{Pulse\ Time}}$$

Return

Interrupt Service Routine (ISR)

Interrupts in Arduino

On a very basic level, an **interrupt is an signal that interrupts the current processor activity**. It may be triggered by an **external event** (change in pin state) or an **internal event** (a timer or a software signal).

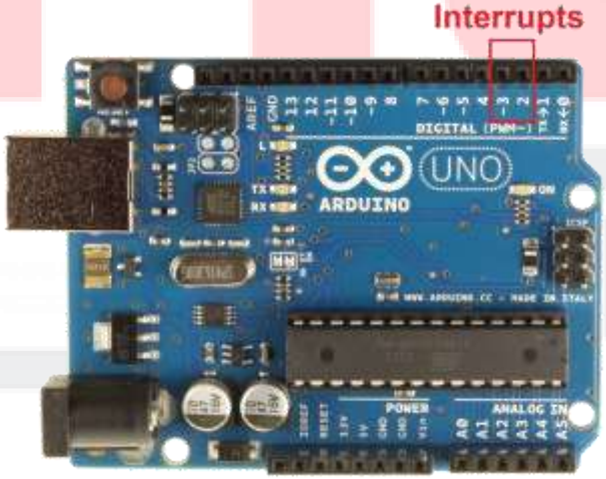


Hardware interrupts: which occur in response to an external event, such as an input pin going high or low (**External Interrupt, Digital Pin**)

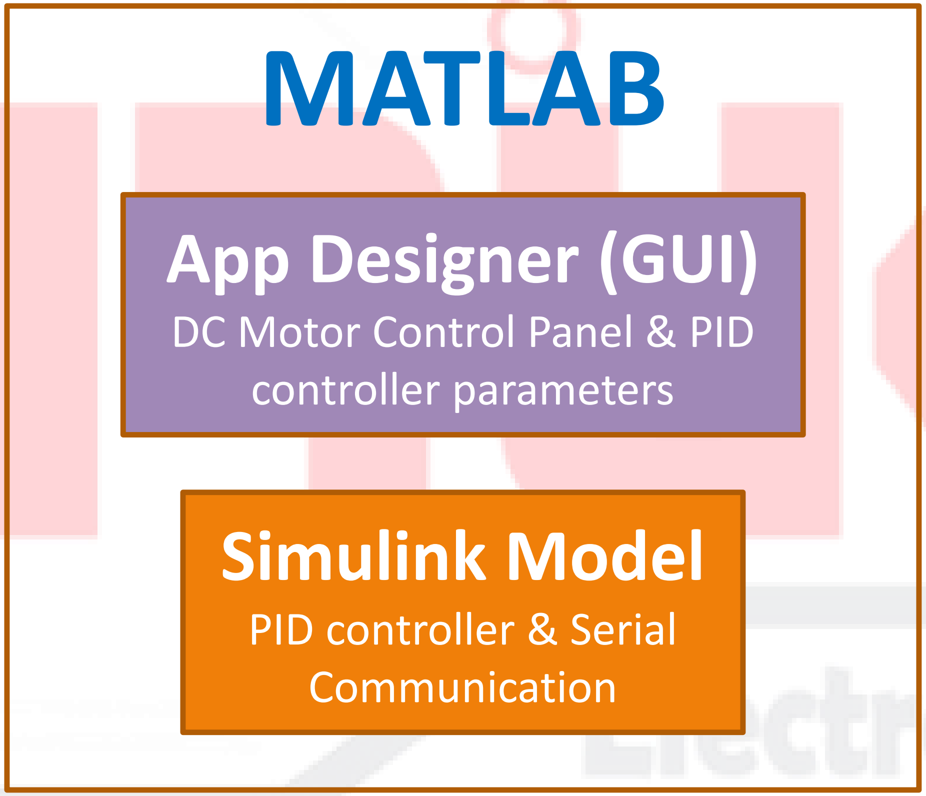
Software interrupts: which occur in response to an instruction sent in software (**Internal Interrupt, Timer**)

Interrupts in Arduino

BOARD	INTERRUPT PINS
Uno, Nano, Mini, other 328-based	2, 3
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21
Micro, Leonardo,	0, 1, 2, 3, 7
Due	all digital pins
101	all digital pins



Computer/MATLAB



MATLAB

App Designer (GUI)

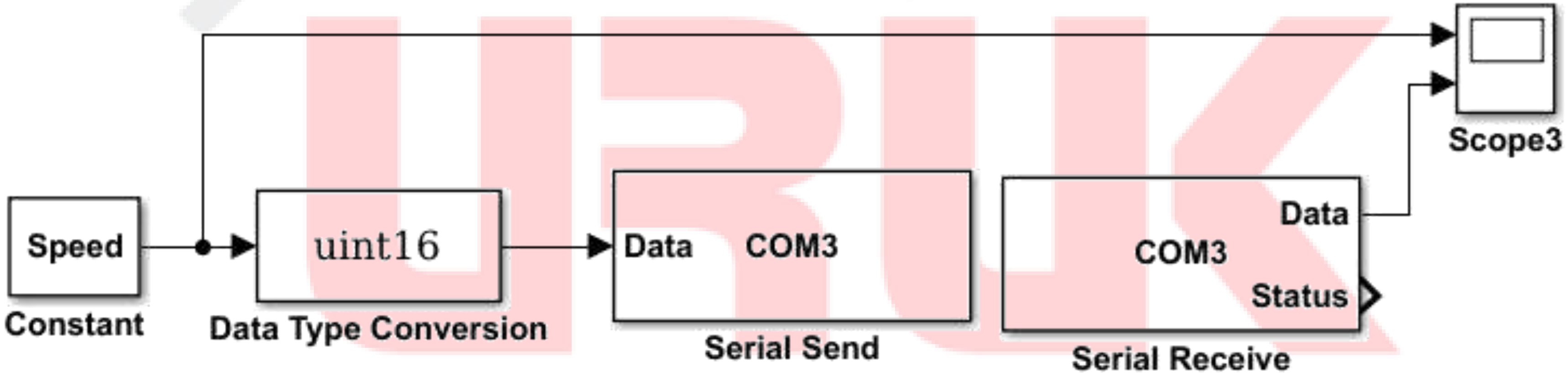
DC Motor Control Panel & PID controller parameters

Simulink Model

PID controller & Serial Communication

Computer/MATLAB

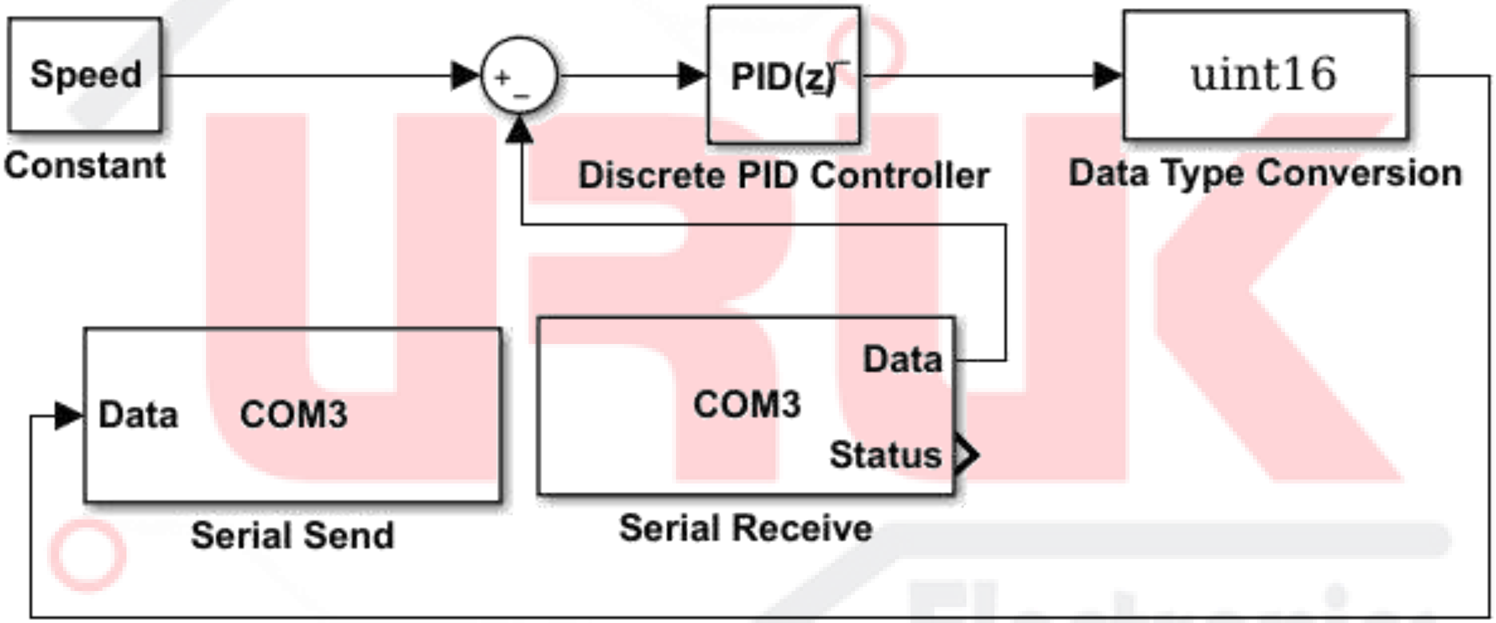
Simulink Model



Open Loop Control Model

Computer/MATLAB

Simulink Model



Closed Loop Control Model

Computer/MATLAB

Simulink Model

Issues presents:

- One of the most common Issues in MATLAB Simulink Serial Interface is that the **MATLAB Simulink doesn't receive Character (ASCII codes).**
- The **motor's speed** is in the range of 4,000 rpm **(integer range value)**, the speed value should be transferred from/to MATLAB via serial port.
- Arduino board's is able to **send Bytes (0-255) only** via its serial !

Computer/MATLAB

Simulink Model

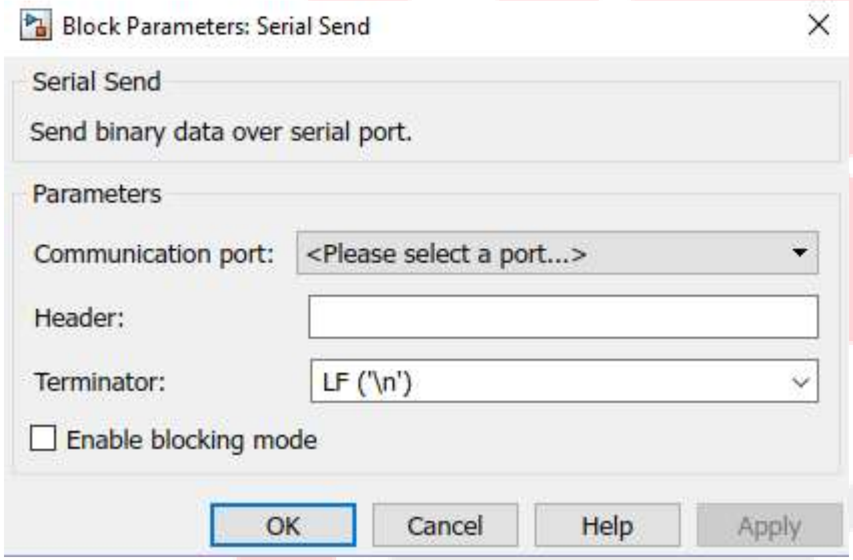
Can be solved by:

- In MATLAB Simulink, the Serial Receive block is set to receive data from Arduino board in Integer format.
- **Developing simple algorithm** in Ardunio to send/receive integer via serial.

Computer/MATLAB

Simulink Model

Simulink's Side



Block Parameters: Serial Send

Serial Send

Send binary data over serial port.

Parameters

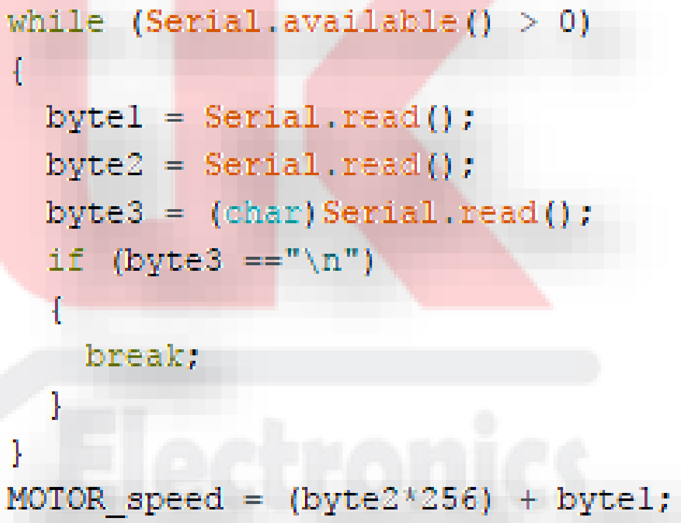
Communication port: <Please select a port...>

Header:

Terminator: LF ('\n')

☐ Enable blocking mode

Arduino Board's Side



```
while (Serial.available() > 0)
{
  byte1 = Serial.read();
  byte2 = Serial.read();
  byte3 = (char)Serial.read();
  if (byte3 =="\n")
  {
    break;
  }
}
MOTOR_speed = (byte2*256) + byte1;
```

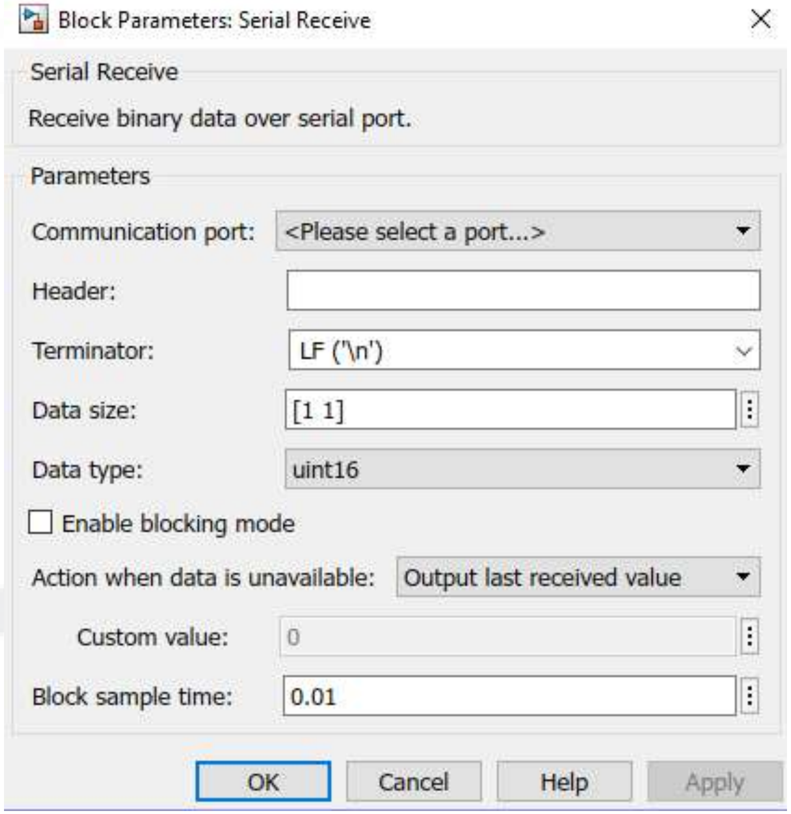
Computer/MATLAB

Simulink Model

Arduino Board’s Side

```
Serial.write(lowByte(rpm));
Serial.write(highByte(rpm));
Serial.print("\n");
```

Simulink’s Side



Block Parameters: Serial Receive

Serial Receive

Receive binary data over serial port.

Parameters

Communication port: <Please select a port...>

Header:

Terminator: LF ('\n')

Data size: [1 1]

Data type: uint16

☐ Enable blocking mode

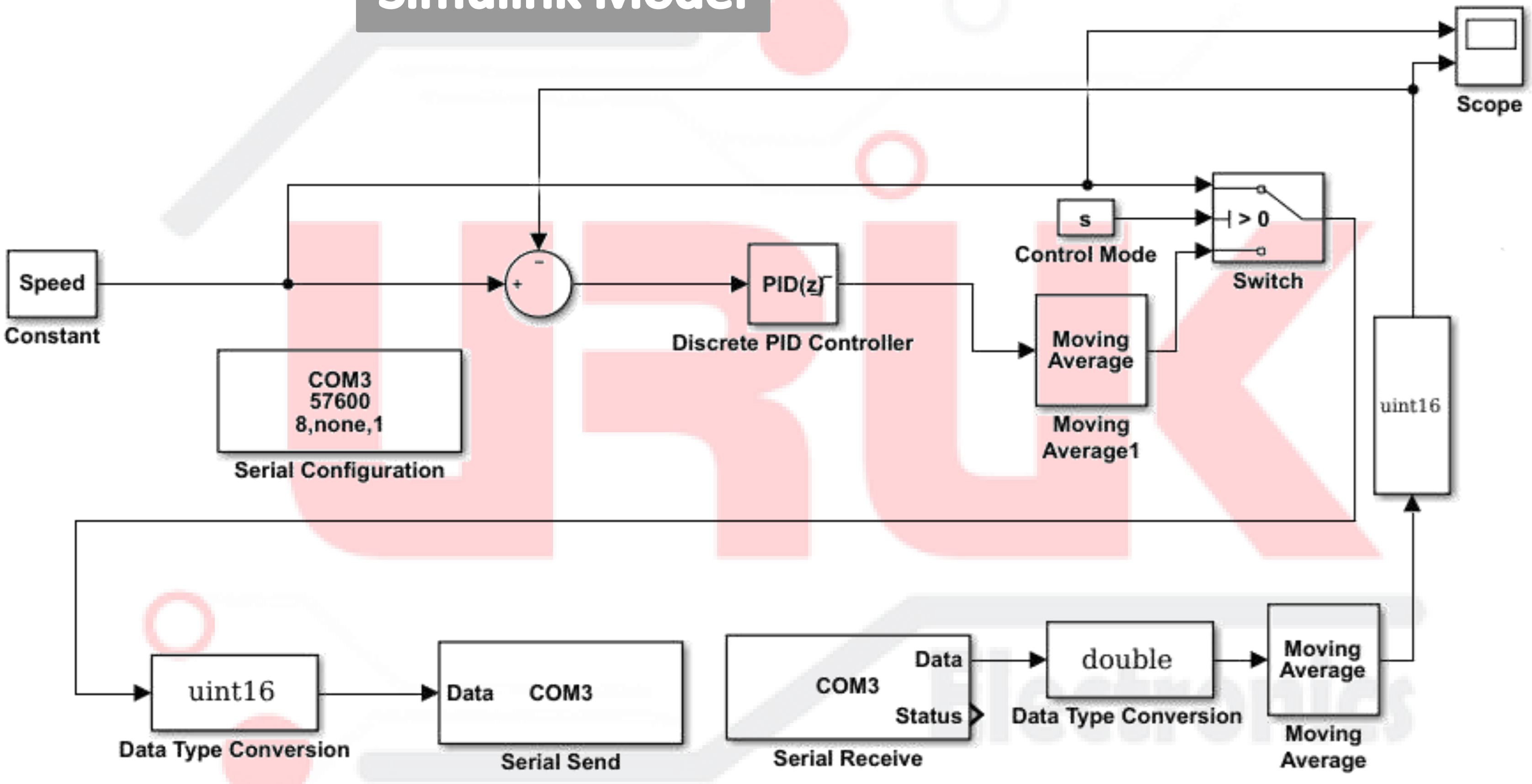
Action when data is unavailable: Output last received value

Custom value: 0

Block sample time: 0.01

Computer/MATLAB

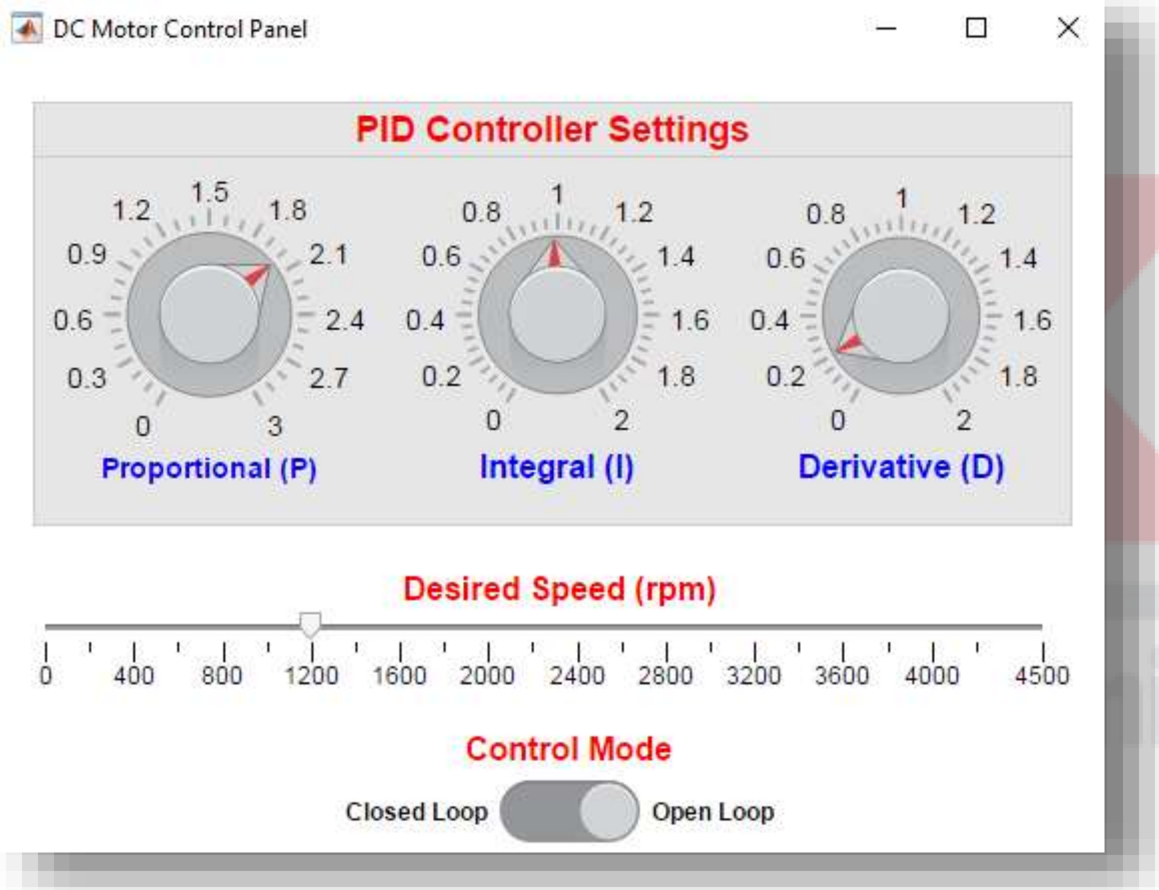
Simulink Model



Simulink's Complete System Model

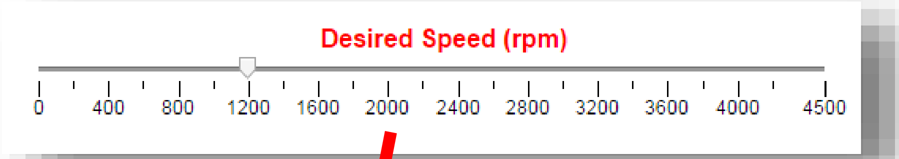
Computer/MATLAB

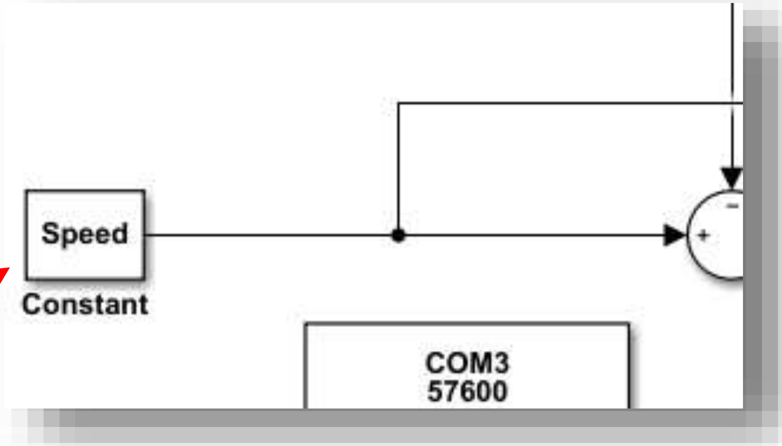
App Designer GUI

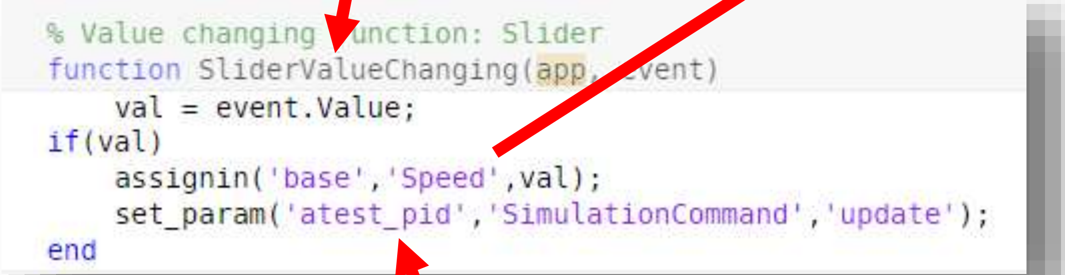


Computer/MATLAB

App Designer GUI







```
% Value changing function: Slider
function SliderValueChanging(app, event)
    val = event.Value;
if(val)
    assignin('base','Speed',val);
    set_param('atest_pid','SimulationCommand','update');
end
```

Simulink file name

Conclusions

- Serial communication is the most popular method for interfacing Peripherals/ Microprocessors within embedded system.
- Arduino Boards can be interfaced as **DAQ system** to any computer software via **serial communication** to form an embedded system.
- The limitation of sending/receiving one byte of data via Serial communication can be overcome by using simple algorithm.
- Controlling a real time system may be govern be any computer software with aid of microcontroller.
- The method used in interfacing the the MATLAB with the Arduino Board (Atmel Microcontroller) could be used for any simulation software with any microcontroller.

Thank you …

Q&A