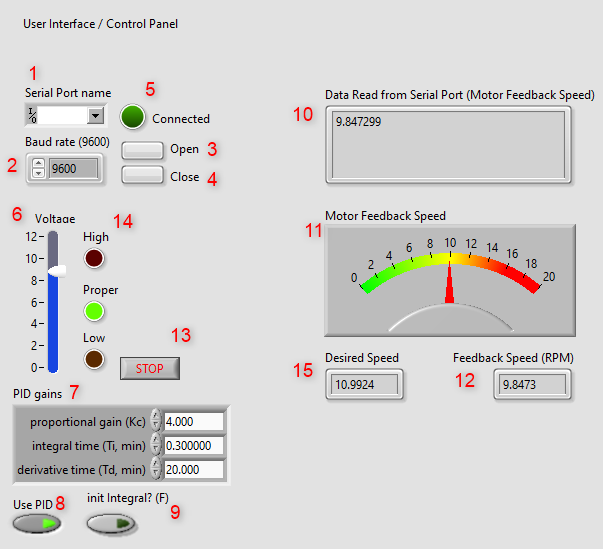
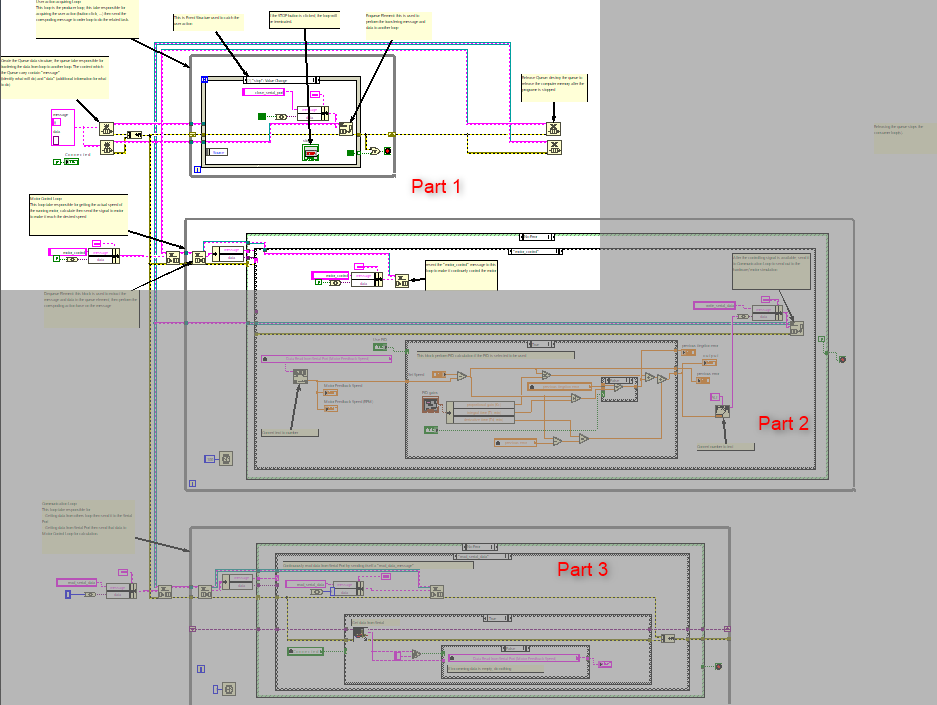
# User Interface for controlling the motor system



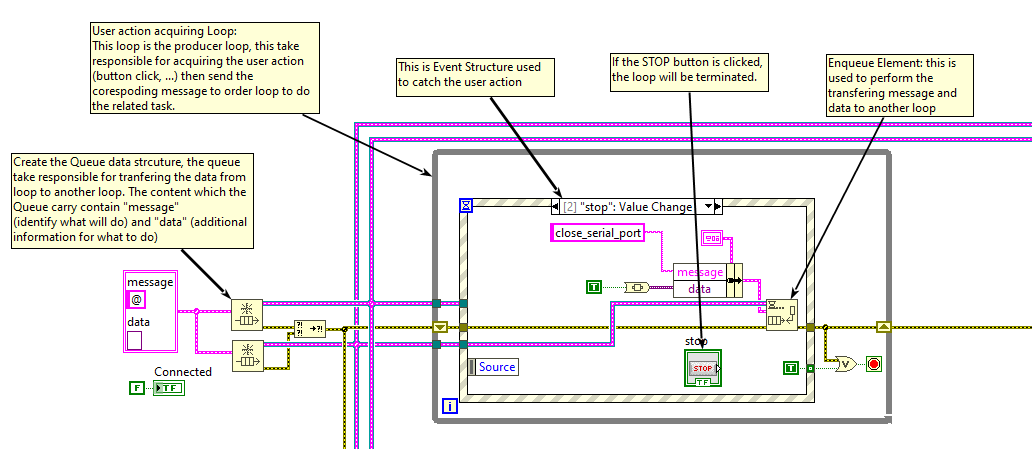
1. Serial Port Name:  
   The name of the Serial Port available for communication (to communicate to the hardware which control the motor or the software which simulate the motor).
2. Baud rate:  
   The baud rate of the Serial port, 9600 is mostly use due to its stable behavior.
3. Open button:  
   Click to open the port and start the communication.
4. Close button :  
   Click to close the port and stop the communication.
5. Connected:  
   The LED to indicate that the communication is ready to be used (send/receive data).
6. Set Speed:  
   The desired motor speed
7. PID gains:  
   Parameter of the PID control
8. Use PID:  
   Turn this button on if would like to use PID, turn off if don’t want to use
9. Init Integral:  
   Reset the Integral value of the PID, this button should be turn on before start using PID, turn it off while using PID
10. Data Read from Serial Port:  
    This box shown the data read from Serial Port (data from hardware which control the motor or the simulation of the motor)
11. Motor Feedback Speed:  
    The value is as same as “Data Read from Serial Port” but view as a Meter Instrument.
12. Motor Feedback Speed (RMP):  
    The value is as same as “Data Read from Serial Port” but view as a Number.
13. STOP button:  
    Used to stop the program
14. Voltage level Indicating LED
15. Desired Speed corresponding to input voltage

**Diagram:**

****

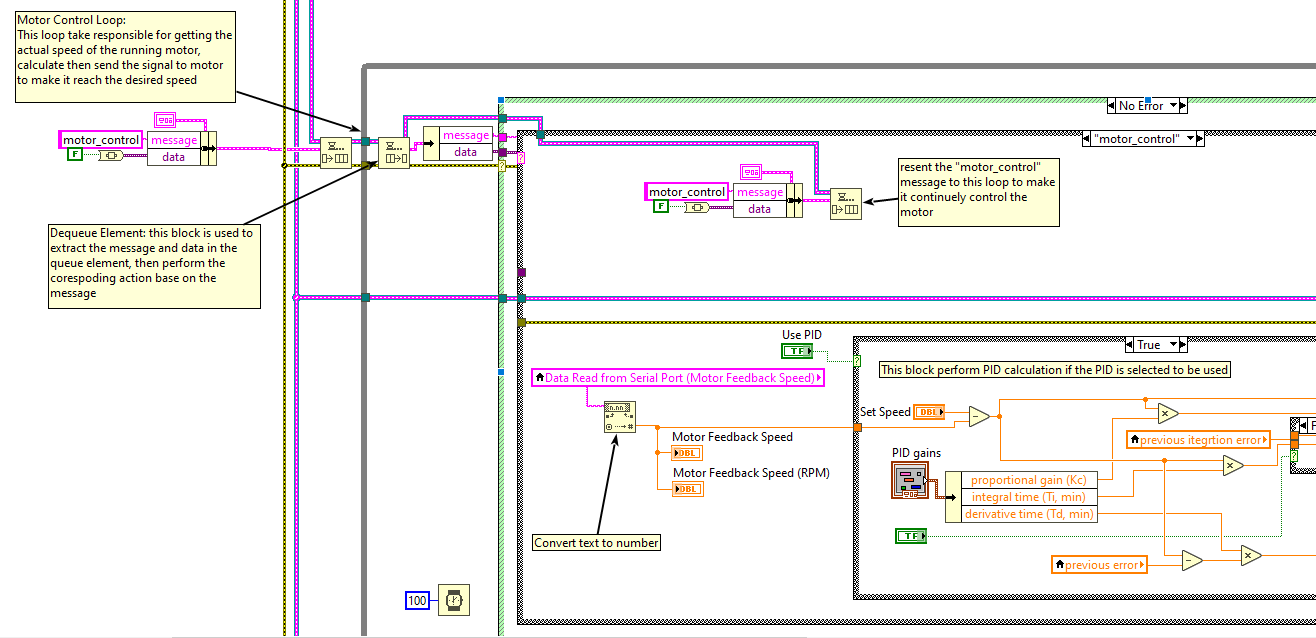
1. **Part 1:**

User action acquiring Loop:  
This loop is the producer loop, this take responsible for acquiring the user action (button click, ...) then send the corespoding message to order loop to do the related task.



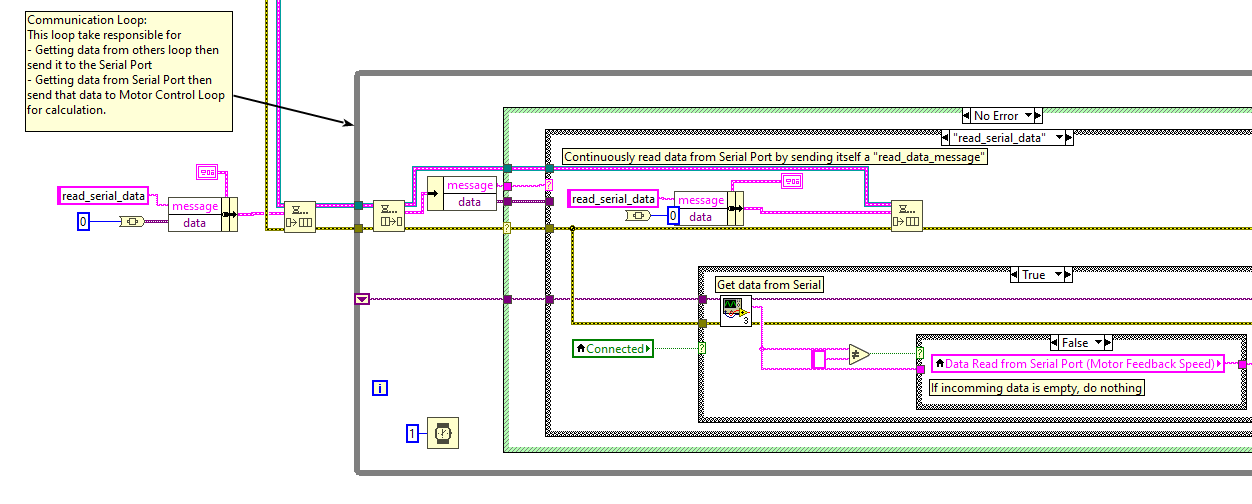
1. **Part 2:**

Motor Control Loop:  
This loop take responsible for getting the actual speed of the running motor, calculate then send the signal to motor to make it reach the desired speed

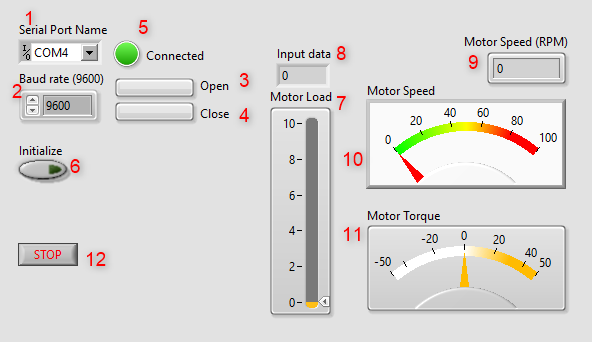


1. **Part 3:**

Communication Loop:  
This loop take responsible for   
- Getting data from others loop then send it to the Serial Port  
- Getting data from Serial Port then send that data to Motor Control Loop for calculation.

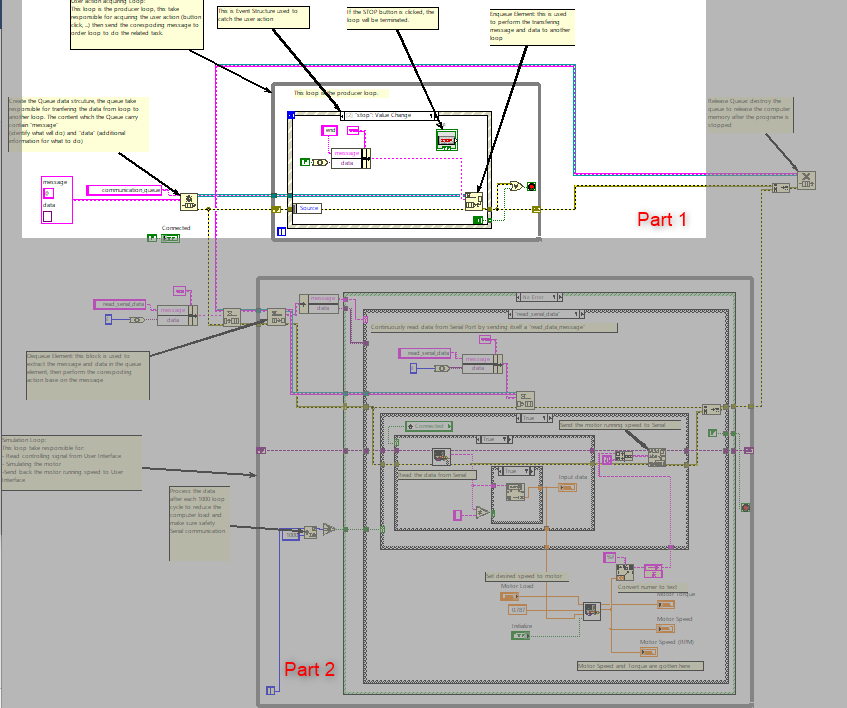
****

# Motor Simulation



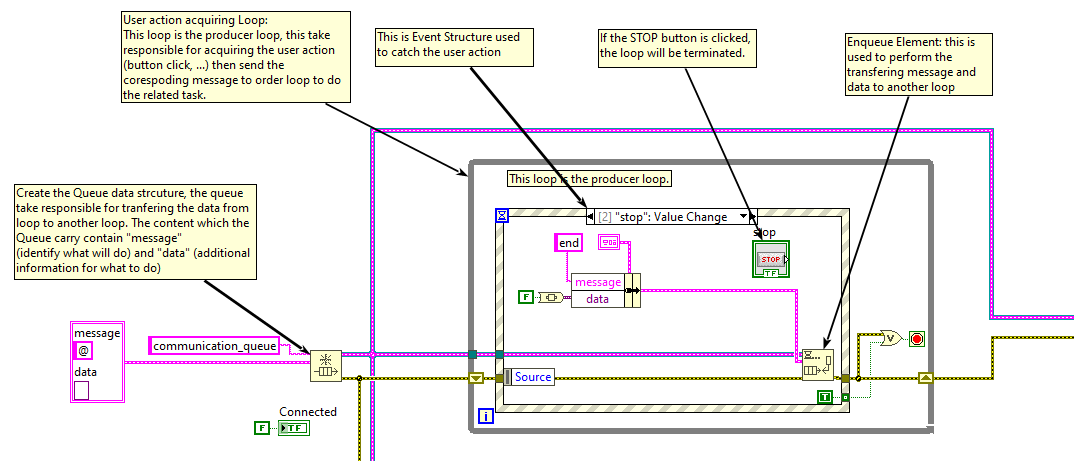
1. Serial Port Name:  
   The name of the Serial Port available for communication (to communicate to the User Interface).
2. Baud rate:  
   The baud rate of the Serial port, 9600 is mostly use due to its stable behavior.
3. Open button:  
   Click to open the port and start the communication.
4. Close button :  
   Click to close the port and stop the communication.
5. Connected:  
   The LED to indicate that the communication is ready to be used (send/receive data).
6. Initialize button:  
   Turn on to reset all internal state of the motor model (speed, torque,… to 0)
7. Motor Load:  
   This bar used to set the load on the motor, set make it default as 0
8. Input data:  
   The data get from Serial Port (this data indicate the requested speed from the user interface)
9. Input Voltage:  
   The motor input voltage corresponding to the desired motor speed.
10. Motor Speed (RMP):  
    The actual motor speed view as a Number
11. Motor Speed:  
    The actual motor speed view as a Meter Instrument
12. Motor Torque   
    The actual torque of the motor
13. STOP button:  
    Click to stop the simulation

**Diagram:**



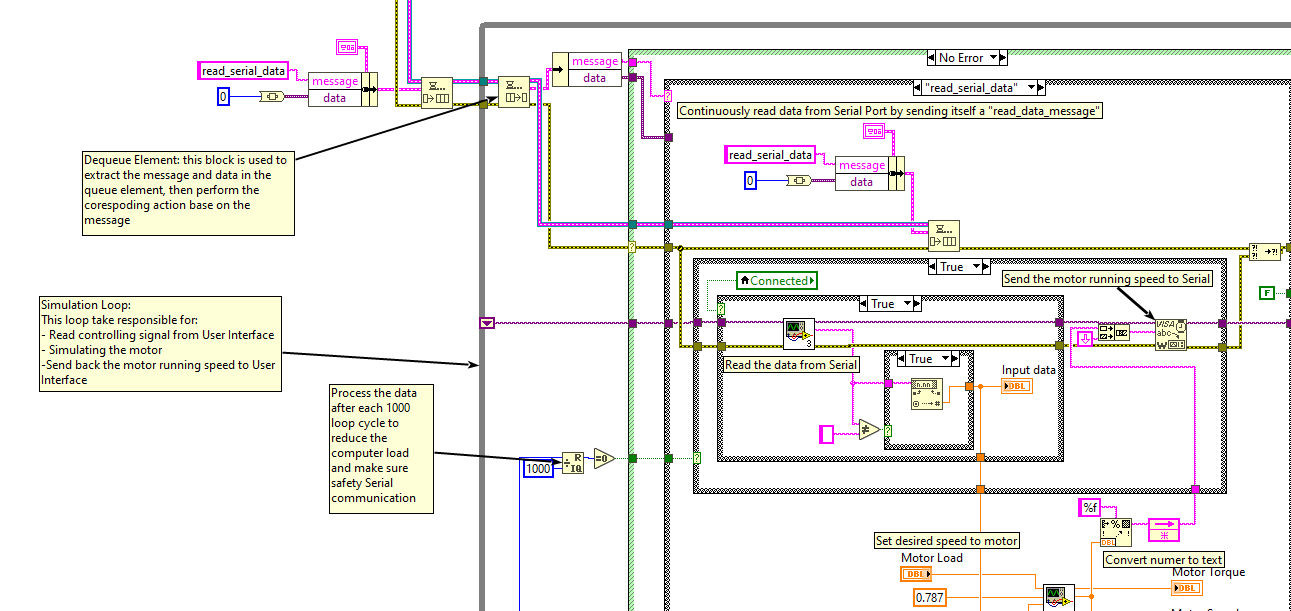
1. **Part 1:**

User action acquiring Loop:  
This loop is the producer loop, this take responsible for acquiring the user action (button click, ...) then send the corespoding message to order loop to do the related task.

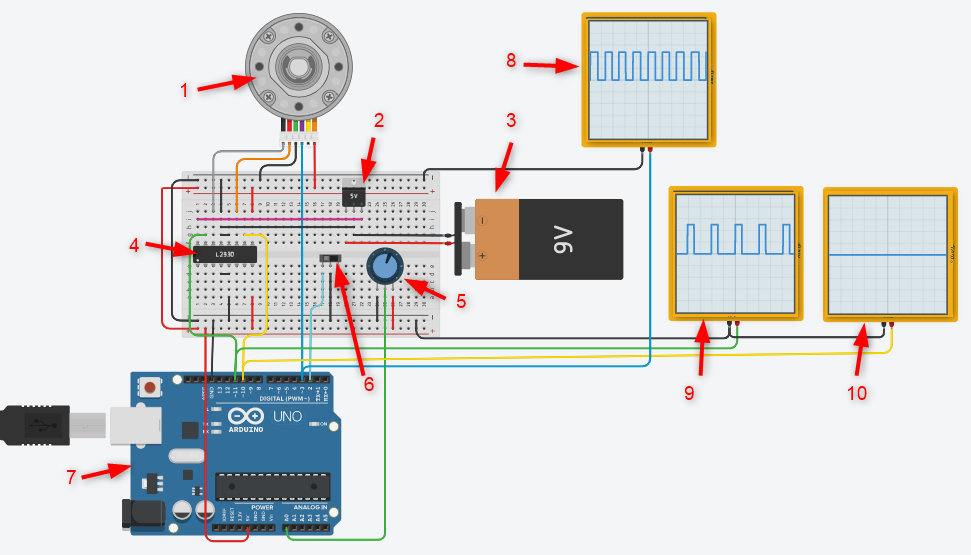


1. **Part 2:**

Simulation Loop:  
This loop take responsible for:   
- Read controlling signal from User Interface  
- Simulating the motor  
-Send back the motor running speed to User Interface



# Tinker CAD Diagram:



1. DC Motor
2. Voltage Regulator
3. Voltage Generator
4. Motor Driver
5. Potential meter for motor speed control
6. Switch to select the motor direction
7. Arduino board
8. Oscilloscope for IR/Encoder Pulse
9. Oscilloscope for Forward PWM pulse
10. Oscilloscope for Reward PWM pulse

# Arduino Code

1. **Assign alias to Arduino pin**

// Macro for pin 1 which control the motor driver

#define MOTOR\_CTRL\_PIN\_1 10

// Macro for pin 1 which control the motor driver

#define MOTOR\_CTRL\_PIN\_2 11

// Macro for the pin read the encoder signal

#define SENSOR\_READ\_PIN 3

// Macro for the pin control the the potentialmeter which control the motor speed

#define POT\_PIN A0

// Macro for the pin control the motor direction

#define DIRECTION\_PIN 2

1. **Assign alias to constants**

//  Macro define motor forward direction

#define TURN\_FORWARD 1

//  Macro define motor reward direction

#define TURN\_REWARD 0

// Macro define the motor speed factor, if the real speed of the motor does not match the desired speed, adjust this value

#define MOTOR\_SPEED\_FACTOR float(2.8)

// Macro define the PWM to RPM

#define PWM\_TO\_RPM\_FACTOR (float(100)/85)

1. **Declare necessary variable**

// Declare variable for speed calculating from the input encorder signal

unsigned int previous\_set\_time = 0;

unsigned int current\_set\_time = 0;

// Declare the variable to store the motor speed

unsigned int motor\_speed = 0;

1. **Calculate Speed function**  
   This function is called every when the SENSOR\_READ\_PIN (Interrput pin on Arduino) change the value from LOW to HIGH), this function measure the time between two RISING edge then calculate the motor speed base on the time between two RISING edge.

/\*

Function to calculate the motor speed from the encoder signal

Each time the encoder signal change from LOW to HIGH, this function is called

\*/

void caculateSpeed(){

  // get the Arduino run time

  current\_set\_time = micros();

  // calculate the speed base on the pulse cycle

  motor\_speed = (float(1000000)/float(current\_set\_time - previous\_set\_time)) / MOTOR\_SPEED\_FACTOR;

  // reset the current time

  previous\_set\_time = current\_set\_time;

}

1. **Motor control function**  
   This function get the parameter dir to decide the direction for the motor, rpm is the desired motor speed.

/\*

Turning motor function, this function control the motor speed and direction

parameter:

  - dir: motor direction

      possible value:

        TURN\_FORWARD

        TURN\_REWARD

  - rpm: motor speed (round per minute)

\*/

void turnMotor(bool dir, int rpm)

{

  analogWrite(MOTOR\_CTRL\_PIN\_1, float(dir) \* float(rpm) \* PWM\_TO\_RPM\_FACTOR);

  analogWrite(MOTOR\_CTRL\_PIN\_2, float(!dir) \* float(rpm) \* PWM\_TO\_RPM\_FACTOR);

}

1. **Setup function for setting up Arduino pin for read/write signal**

/\*

void setup() funtion

The Arduino void setup and void loop functions are mandatory.

When you run a “standard” C/C++ program, you have to write a “main” function.

This main function will be called first, and from there,

you will call other functions and execute the functionalities of your program.

\*/

void setup()

{

  // setup the motor control pin 1 as OUTPUT

  pinMode(MOTOR\_CTRL\_PIN\_1, OUTPUT);

  // setup the motor control pin 2 as OUTPUT

  pinMode(MOTOR\_CTRL\_PIN\_2, OUTPUT);

  // setup the encoder read as INPUT\_PULLUP

  pinMode(SENSOR\_READ\_PIN, INPUT\_PULLUP);

  // setup the input direction switch as INPUT\_PULLUP

  pinMode(DIRECTION\_PIN, INPUT\_PULLUP);

  // Set up the pin which connect to encoder as an interrput catch the RISING edge

  attachInterrupt(digitalPinToInterrupt(SENSOR\_READ\_PIN), caculateSpeed, RISING);

  // setup the Serial port

  Serial.begin(9600);

  // initialize the time value for motor speed calculation

  previous\_set\_time = micros();

  current\_set\_time = micros();

  // Start the motor

  turnMotor(TURN\_REWARD, 100);

}

1. **Loop function for executing the Motor control and print out the system information (Control mode, motor direction, motor speed) to Serial Monitor.**

/\*

In Arduino, there is no main function. This is replaced by setup and loop

The code inside the void loop will be executed again and again (hence the name “loop”), until you:

  - Power off the Arduino board.

  -or Restart the Arduino program by

    pressing the reset button /

    uploading a new sketch /

    re-opening the Serial Monitor on some Arduino boards.

\*/

void loop()

{

  // Read the direction signal

  bool direction = digitalRead(DIRECTION\_PIN);

  // Get the motor speed base on potentialmeter signal

  int speed = analogRead(POT\_PIN)/10;

  // Control the motor from direction and speed gotten above

  turnMotor(direction, speed);

  // Print out the motor state to Serial monitor

  // if the motor direction is forward

  if(direction == TURN\_FORWARD)

  {

    // print the text TURN\_FORWARD

    Serial.print("TURN\_FORWARD: ");

  }

  // if the motor direction is reward

  else if(direction == TURN\_REWARD)

  {

    // print the text TURN\_REWARD

    Serial.print("TURN\_REWARD: ");

  }

  // print the motor speed

  Serial.print("\t");

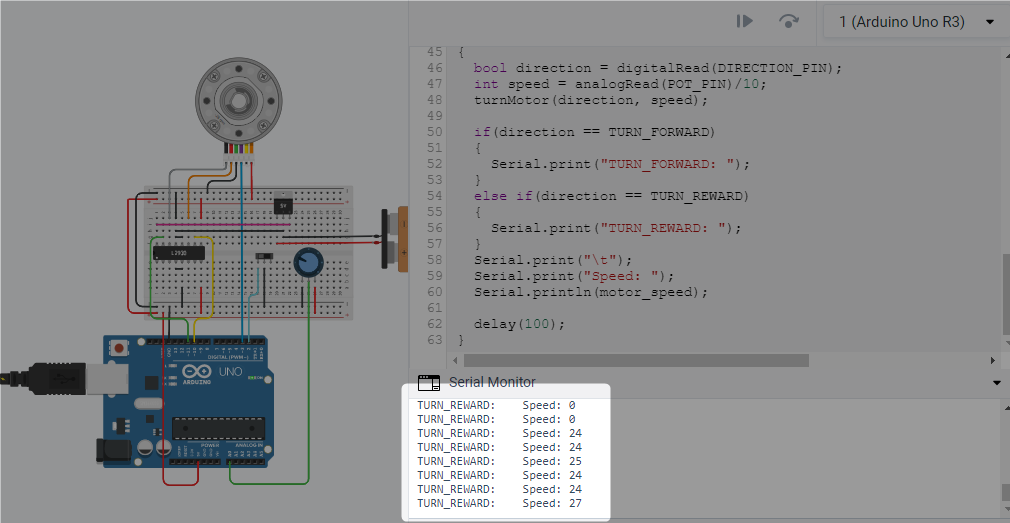
  Serial.print("Speed: ");

  Serial.println(motor\_speed);

  // delay 100ms until the next cycle

  delay(100);

}



**Overview** **Diagram:**

**Prj1\_Main.vi**

Port

Motor control loop  
- PID  
- PID parameter  
- Feed back Speed  
- Feed back Speed (as number)

Serial Bus

Real running speed

Desired Speed

**System\_1.vi (motor simulation)**

Port

Simulation Loop

Motor Simulation

Serial port control

User interface acquiring loop

Communication Loop

User interface acquiring loop