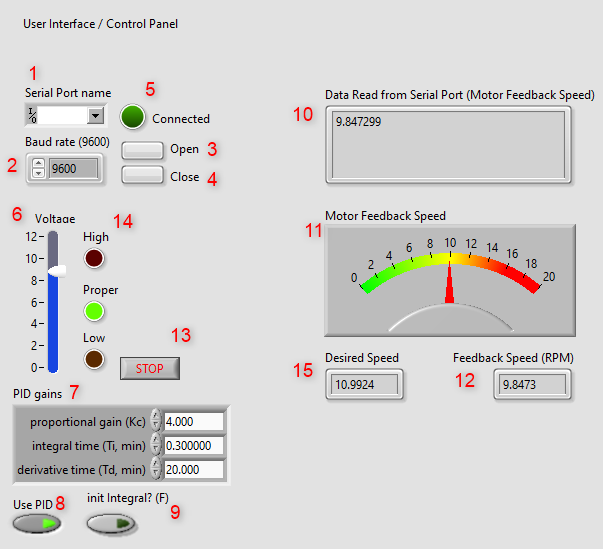
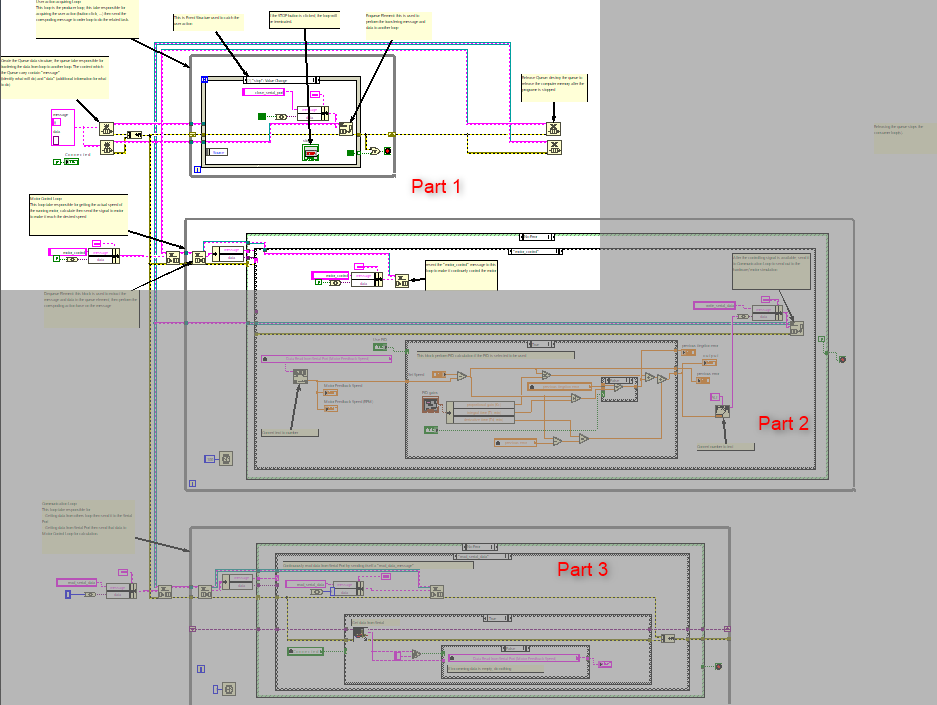
# User Interface for controlling the motor system



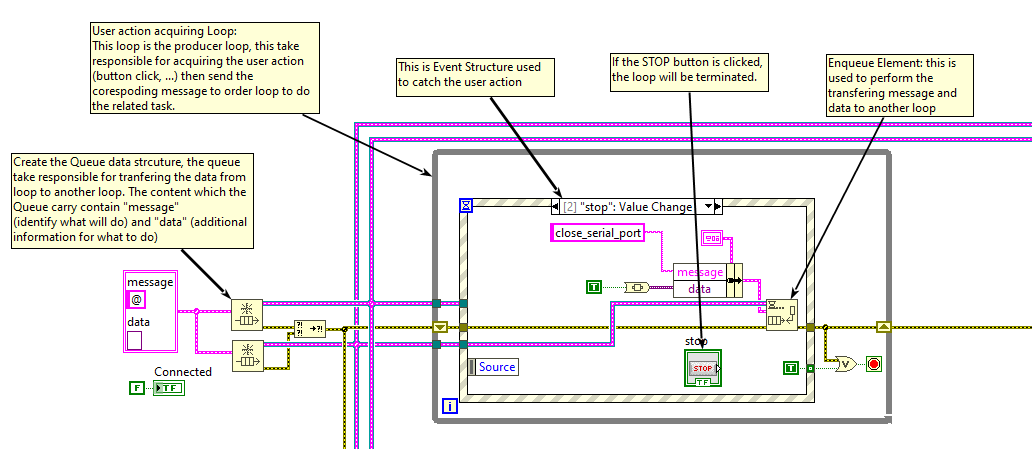
1. Serial Port Name:  
   The name of the Serial Port available for communication (to communicate to the hardware which control the motor or the software which simulate the motor).
2. Baud rate:  
   The baud rate of the Serial port, 9600 is mostly use due to its stable behavior.
3. Open button:  
   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:**

****

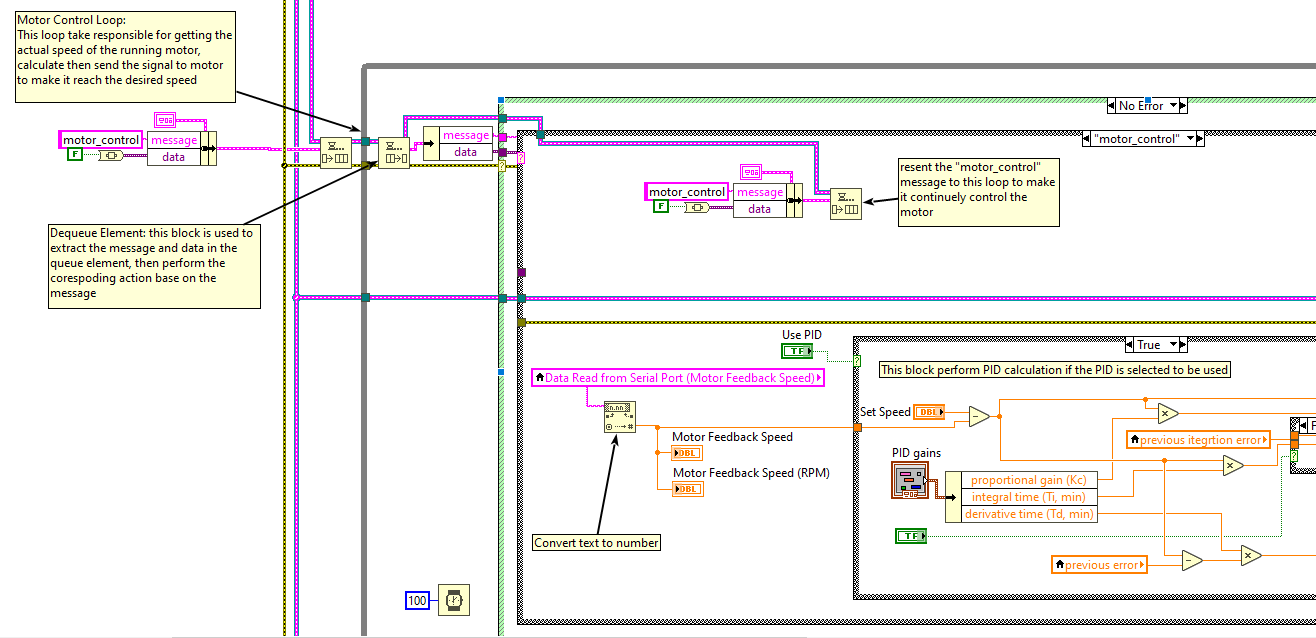
1. **Part 1:**

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.



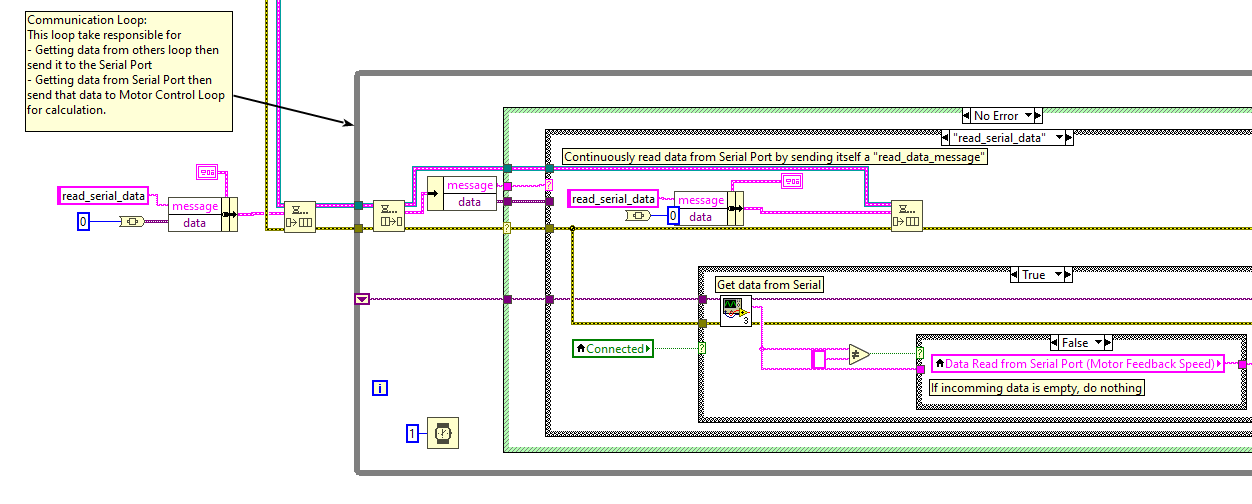
1. **Part 2:**

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

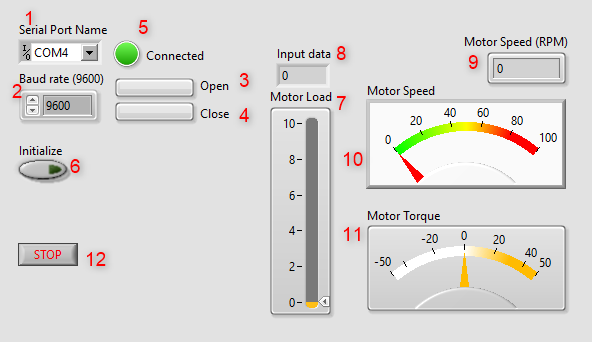


1. **Part 3:**

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.

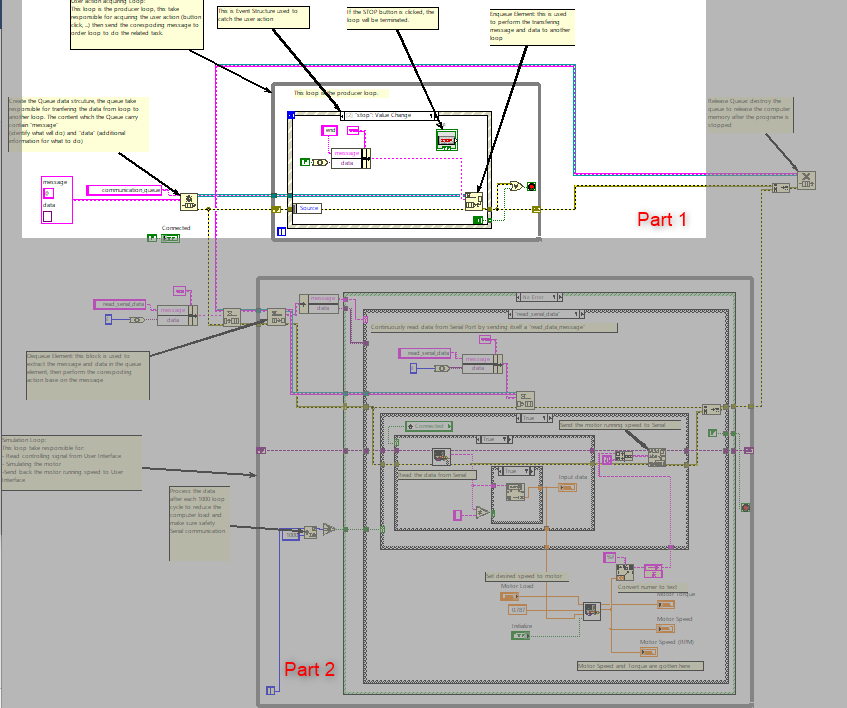
****

# Motor Simulation



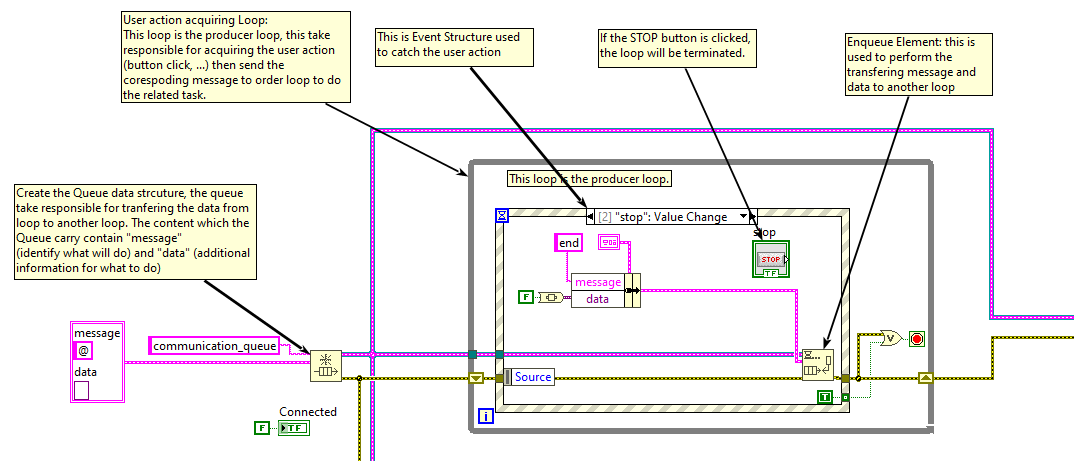
1. Serial Port Name:  
   The name of the Serial Port available for communication (to communicate to the User Interface).
2. Baud rate:  
   The baud rate of the Serial port, 9600 is mostly use due to its stable behavior.
3. Open button:  
   Click to open the port and start the communication.
4. Close 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:**



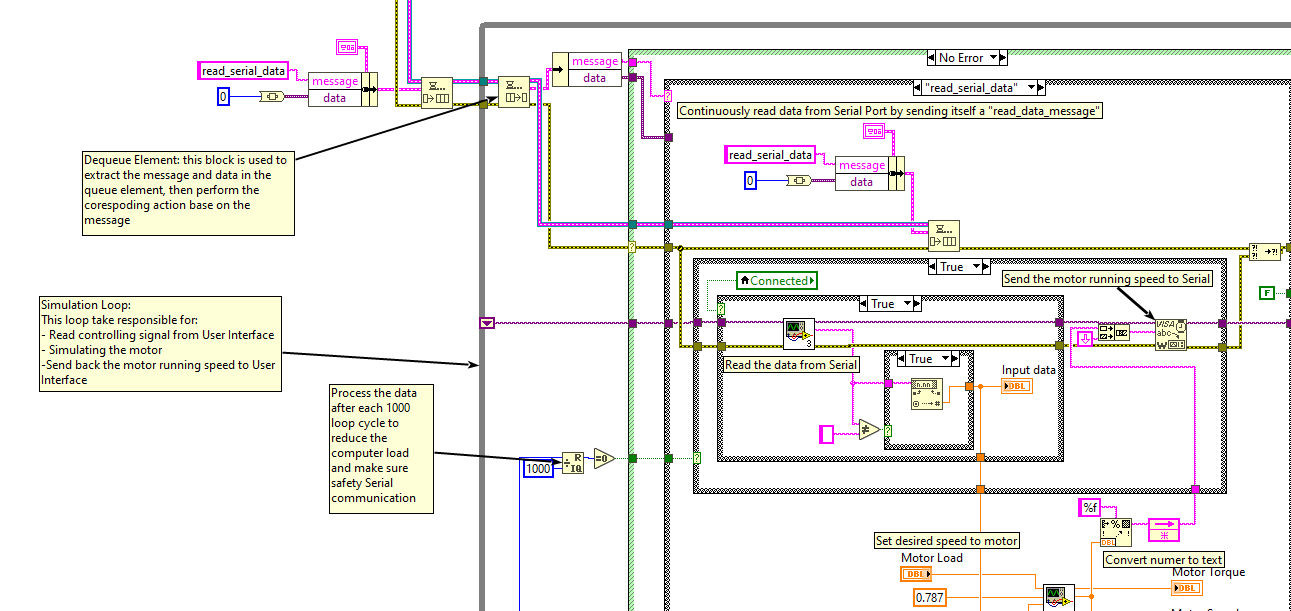
1. **Part 1:**

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.

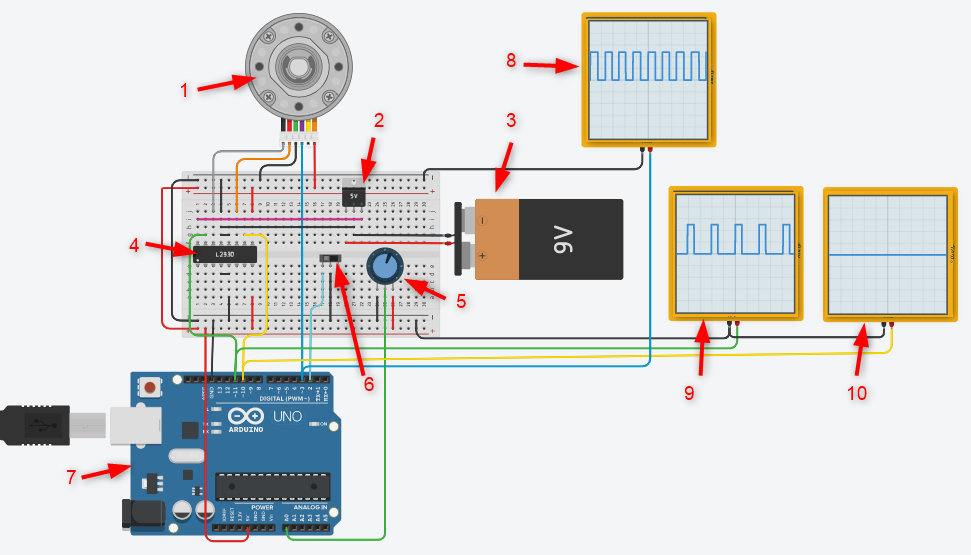


1. **Part 2:**

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**

#define MOTOR\_CTRL\_PIN\_1 10

#define MOTOR\_CTRL\_PIN\_2 11

#define TURN\_FORWARD 1

#define TURN\_REWARD 0

#define SENSOR\_READ\_PIN 3

#define POT\_PIN A0

#define DIRECTION\_PIN 2

1. **Assign alias to constants**

#define MOTOR\_SPEED\_FACTOR float(2.8)

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

1. **Declare necessary variable**

unsigned int previous\_set\_time = 0;

unsigned int current\_set\_time = 0;

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.

void caculateSpeed(){

  current\_set\_time = micros();

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

  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.

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()

{

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

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

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

    pinMode(DIRECTION\_PIN, INPUT\_PULLUP);

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

    Serial.begin(9600);

    previous\_set\_time = micros();

    current\_set\_time = micros();

    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.**

void loop()

{

  bool direction = digitalRead(DIRECTION\_PIN);

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

  turnMotor(direction, speed);

  if(direction == TURN\_FORWARD)

  {

    Serial.print("TURN\_FORWARD: ");

  }

  else if(direction == TURN\_REWARD)

  {

    Serial.print("TURN\_REWARD: ");

  }

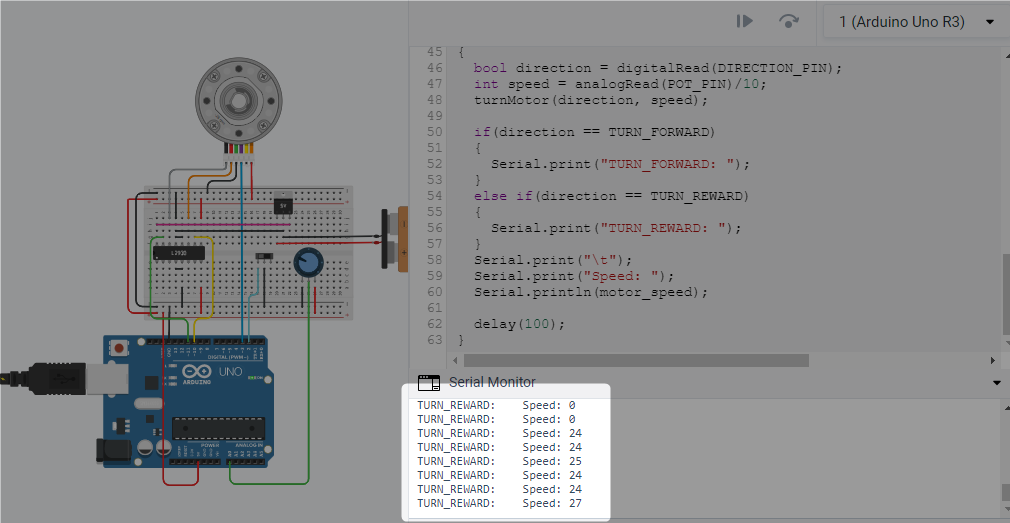
  Serial.print("\t");

  Serial.print("Speed: ");

  Serial.println(motor\_speed);

  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