# **Wire Prepping Machine Design Report**

## **1. Introduction**

In this project, I designed a wire prepping machine for handling a 5.5mm diameter, 4-core wire. The machine will perform various operations, such as cutting of wire, removing the external sheathing, stripping the cores, and tinning the cores. The design will be controlled via a local web portal (i.e. we can use XYremote software) and will feature error-handling mechanisms displayed both on the machine and the web interface. The project focuses on control system design, automation, and integration of sensors and actuators, with the mechanical design being handled separately.

## **2. Objectives and Design Requirements**

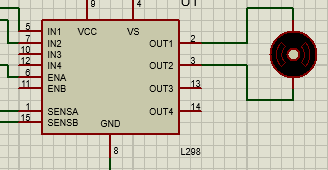
The wire-prepping machine design aims to achieve the following objectives:

1. ***Wire Handling:*** The machine will handle 5.5mm diameter, 4-core wire, which is continuously fed from an endless roll.
2. ***Cutting the wire:*** The machine will also be responsible for cutting the wire in a programmable fashion.
3. ***Sheathing Removal:*** The machine will remove the external sheathing on both ends of the wire, with programmable lengths for processing.
4. ***Core Stripping and Tinning:*** The machine will strip and tin the cores on both ends of the wire, also with programmable lengths.
5. ***Machine Operation:*** The machine can be operated via a local web portal to start and stop tasks.
6. ***Error Reporting:*** Errors in operation will be displayed both on the machine’s screen (Oled or 7-segment display) and on the web portal. Additionally, dry contact points will indicate errors.
7. ***Runout Sensor:*** A sensor will detect the wire’s runout to prevent malfunction (limit switch).

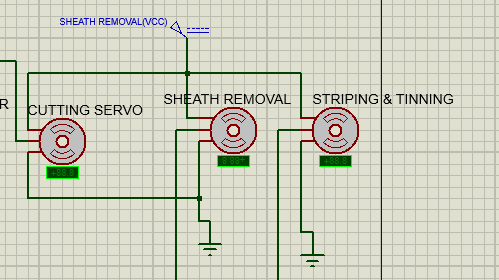
## **3. System Components and Architecture**

### ****3.1. Actuators and Sensors****

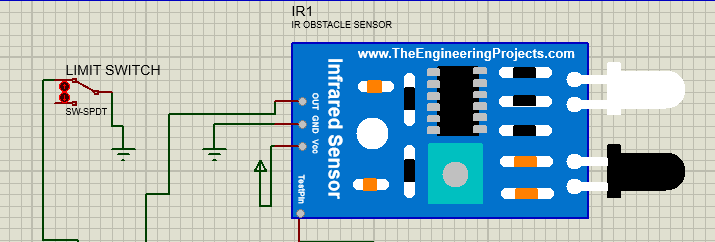
* **DC Motors**: Used for feeding the wire, moving it forward, and operating the motorized mechanisms for wire feeding.



* **Servo motors**: Controls the cutting of the wire, sheathing removal mechanism, and stripping/tinning of the cores.

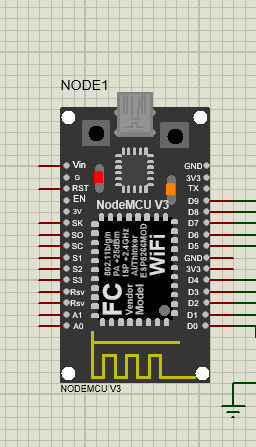


* **Sensors**:
  + **Runout Sensor**: a *limit switch* detects the end of the wire to halt operations.
  + **IR Sensor**: Detects the wire length as it moves through the DC motor.



### ****3.2. Control Unit****

* **NodeMCU V3 ESP8266**: Handles control of the motors, servos, sensors, and web portal interface. It will manage all operations.



## **4. Operational Flow and Task Execution**

*The wire-prepping machine will follow the following steps:*

**1. Wire Detection**

**Limit Switch Activation**: The wire is continuously fed through the system from the roll. The **limit switch** will detect the initial position of the wire. Once the wire is in position, the system is ready to begin feeding.

**2. Feeding the Wire**

**DC Motor Operation**: The **DC motor** will draw the wire forward. The wire will move through the system, and the **IR sensor** will be used to measure the length of wire fed.

* + **IR Sensor Role**: The **IR sensor** counts the rotations of the DC motor. For instance, one complete rotation of the motor corresponds to 5 cm of wire being fed forward.
  + **Length Calculation**: When the DC motor completes two full rotations (i.e., 10 cm), the system recognizes that the correct length of wire has been fed.

**3. Wire Cutting**

**Servo Motor for Cutting**: Once 10 cm of wire has been fed, a **servo motor** attached to the cutting mechanism will activate. The servo will cut the wire into the required segment length.

### ****4. Sheathing Removal****

**Servo Motor for Sheathing Removal**: After cutting, the wire will be moved forward to a **servo motor** responsible for sheathing removal. This servo will strip the external sheathing of both ends of the wire. The length of the sheathing to be removed will be programmable, allowing for flexibility.

### ****5. Core Stripping and Tinning****

* **Motorized Core Stripping Mechanism**: The wire will then move to the next step, where another **motorized mechanism** grabs the cores inside the wire.
* **Servo Motor for Stripping**: The **servo motor** will strip the wire cores, preparing them for tinning.
* **Soldering**: A **soldering head** will be positioned over the wire cores to tin them with solder. The soldering head will be moved to each core as required.

**6. Error Handling**

If there are issues such as wire runout, or if a mechanism fails, the machine will display an error message on both the machine's screen and the local web portal. Additionally, LED will indicate the error externally.

* And the process will repeat again and again.

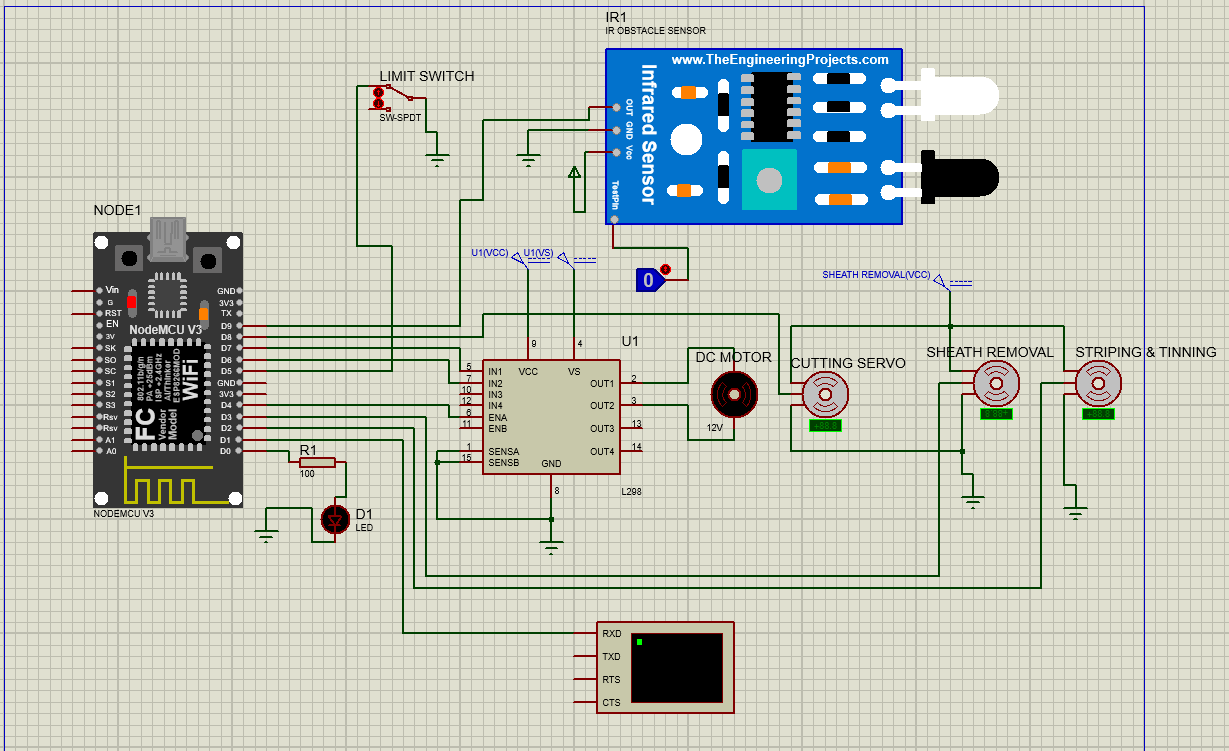


Figure 2: Wire Prepping Machine Design 1

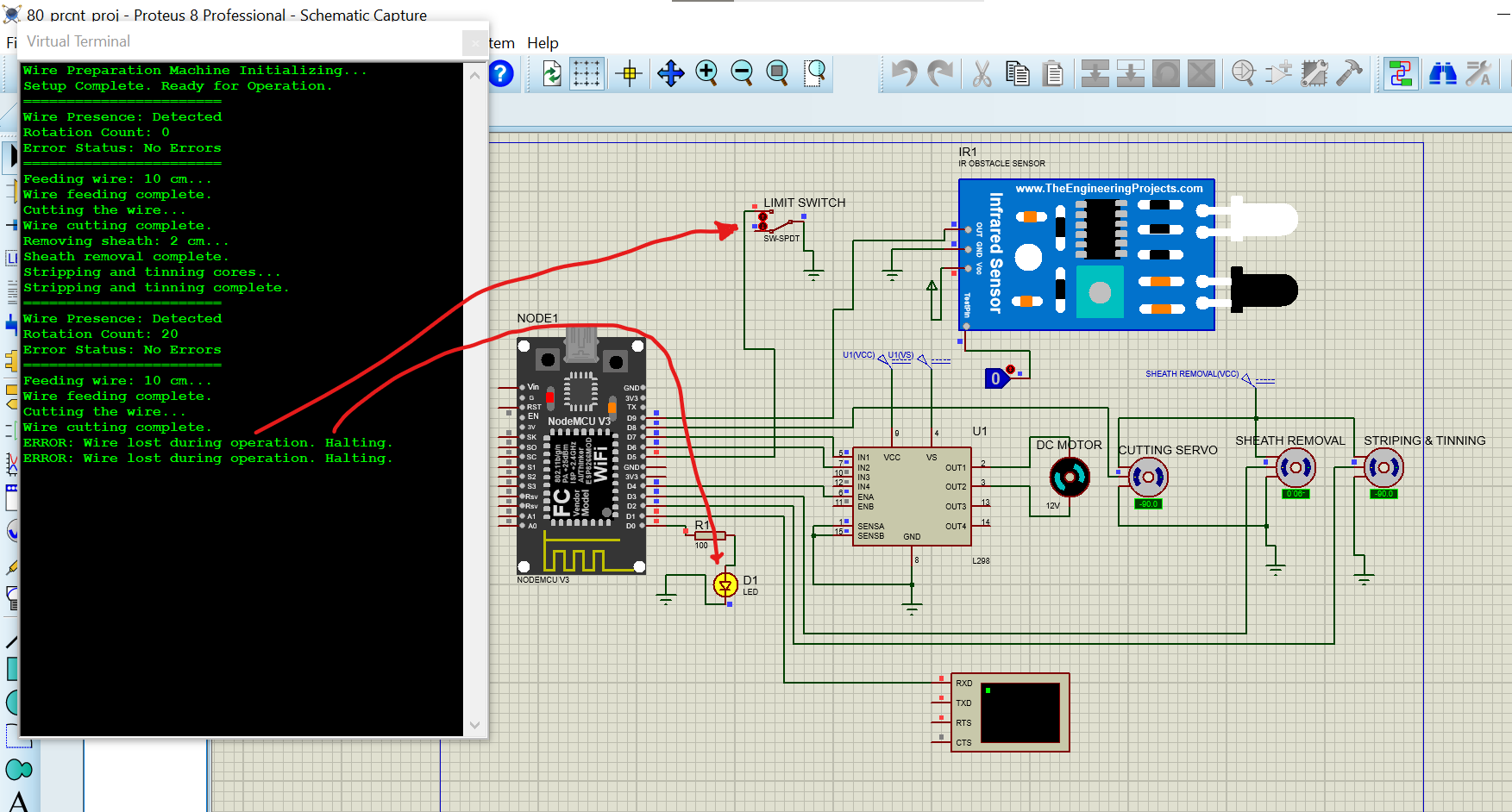


Figure 1: Result on virtual terminal

## **5. Design and Control System**

The control system of the wire-prepping machine includes both the hardware (motors, servos, sensors, displays, LED’s) and software (Arduino IDE).

### ****5.1. Hardware Design****

The following components are used:

* ***DC Motors*** for wire feeding.
* ***Display(Oled or 7-segment)***
* ***Servo motors*** for cutting, sheathing removal, and stripping/tinning.
* ***NodeMCU ESP8266*** as the central controller.
* ***LED’s for error indications***
* ***Sensors*** (runout 🡪 limit switch, wire length sensor 🡪 IR sensor, and error sensors) for operation and error detection.

### ****5.2. Software Design****

The machine’s software will be written using Arduino IDE, with the following functionalities:

* ***Web Interface:*** A local web portal will control the machine’s actions, including starting/stopping tasks, setting wire lengths, and displaying the current machine status.
* ***Motor and Servo Control:*** The DC motors and servos will be controlled to perform wire feeding, cutting, sheathing removal, and core tinning in sequence.
* ***Error Handling:*** The system will continuously check for errors such as wire runout or malfunctioning mechanisms and report them to the web portal and machine display.

## **6. Assumptions and Limitations**

### ****6.1. Assumptions****

* The wire is continuously fed from an endless roll, and no custom feeder system is required.
* The machine will handle only 5.5mm diameter, 4-core wires.
* External mechanical design for wire handling (cutting, stripping) will be handled by mechanical design team.
* The local web portal is only for local operation and monitoring & remote access are not included in this design.
* The Wifi portal will be set up properly as required in real-time.
* In real-time the display will be properly configured with esp8266.

### ****6.2. Limitations****

* In proteus simulation the wifi doesn't work as expected, that’s why web portal or wifi capabilities in proteus were not set up properly.
* The proper display is not set up properly because of proteus limitations.
* Precision in wire length depends on motor rotation calibration.

## **7. Code Implementation**

The following is the basic code implementation that will control the motors and servos, and handle web interface interactions:

|  |
| --- |
| #include <Servo.h>  #include <ezButton.h>  // #include <ESP8266WiFi.h>  // #include <ESP8266WebServer.h>  // Define pins for peripherals  #define LIMIT\_SWITCH\_PIN 5   // Limit switch to detect wire runout  #define IR\_SENSOR\_PIN 9      // IR sensor for rotation count  #define DC\_MOTOR\_IN1 7       // DC motor IN1  #define DC\_MOTOR\_IN2 6       // DC motor IN2  #define DC\_MOTOR\_ENA 4       // DC motor enable (PWM control)  #define SERVO\_SHEATH 3       // Servo for sheathing removal  #define SERVO\_SOLDER 2       // Servo for soldering  #define SERVO\_CUTTER 8       // Servo for wire cutting  #define ERROR\_LED 0          // LED for error indication  //#define ERROR\_CONTACT\_PIN 10 // Dry contact for external error reporting  // WiFi credentials  //const char\* ssid = "Your\_SSID";  //const char\* password = "Your\_PASSWORD";  // Servo objects  Servo servoSheath;  Servo servoSolder;  Servo servoCutter;  // Button object for the limit switch  ezButton limitSwitch(LIMIT\_SWITCH\_PIN);  // Variables for system state  volatile int rotationCount = 0;  // Count of rotations detected by the IR sensor  bool wirePresent = false;        // Wire presence status  bool errorFlag = false;          // System error state  bool taskRunning = false;        // Task in progress  // Configurable parameters  int wireLength = 10;          // Wire length in cm (can be adjusted via web input)  int sheathLength = 2;         // Sheathing removal length in cm  int rotationsPerCm = 2;       // IR sensor rotations per cm of wire  // ESP8266 Web Server  // ESP8266WebServer server(80);  void setup() {    // Initialize Serial communication    Serial.begin(9600);    Serial.println("Wire Preparation Machine Initializing...");    // Limit switch debounce setup    limitSwitch.setDebounceTime(50);    // Initialize IR sensor pin    pinMode(IR\_SENSOR\_PIN, INPUT);    // Initialize DC motor pins    pinMode(DC\_MOTOR\_IN1, OUTPUT);    pinMode(DC\_MOTOR\_IN2, OUTPUT);    pinMode(DC\_MOTOR\_ENA, OUTPUT);    // Initialize error LED    pinMode(ERROR\_LED, OUTPUT);    //pinMode(ERROR\_CONTACT\_PIN, OUTPUT); // Optional dry contact for error    // Attach servo motors    servoSheath.attach(SERVO\_SHEATH);    servoSolder.attach(SERVO\_SOLDER);    servoCutter.attach(SERVO\_CUTTER);    // Reset components    stopDCMotor();    servoSheath.write(0);    servoSolder.write(0);    servoCutter.write(0);    // Setup WiFi    //WiFi.begin(ssid, password);    //while (WiFi.status() != WL\_CONNECTED) {    //  delay(500);    //  Serial.print(".");    //}    //Serial.println("\nWiFi connected");    //Serial.print("IP Address: ");    //Serial.println(WiFi.localIP());    // // Configure Web Server routes    // server.on("/", handleRoot);    // server.on("/set", handleSetParameters);    // server.begin();    // Serial.println("Web server started.");    Serial.println("Setup Complete. Ready for Operation.");  }  void loop() {    // Update limit switch state    limitSwitch.loop();    // Check for wire presence    wirePresent = limitSwitch.getState() == LOW;    // Handle web server requests    // server.handleClient();    // Display status on Serial Monitor    displayStatus();    // Handle errors if wire is not detected    if (!wirePresent) {      handleError("Wire not detected. System halted.");      return;    } else {      clearError();    }    // If no errors and not already running, start task    if (!taskRunning) {      taskRunning = true;      // Execute the steps in sequence      if (checkWirePresence()) feedWire(wireLength);      if (checkWirePresence()) cutWire();      if (checkWirePresence()) removeSheath(sheathLength);      if (checkWirePresence()) stripAndTinCores();      taskRunning = false; // Mark task complete    }    // Pause briefly before next iteration    delay(1000);  }  // Check wire presence during operation  bool checkWirePresence() {    limitSwitch.loop();    wirePresent = limitSwitch.getState() == LOW;    if (!wirePresent) {      handleError("Wire lost during operation. Halting.");      return false;    }    return true;  }  // Feed wire by rotating the motor  void feedWire(int length) {    Serial.print("Feeding wire: ");    Serial.print(length);    Serial.println(" cm...");    rotationCount = 0; // Reset rotation count    analogWrite(DC\_MOTOR\_ENA, 200); // Set motor speed    digitalWrite(DC\_MOTOR\_IN1, HIGH);    digitalWrite(DC\_MOTOR\_IN2, LOW);    while (rotationCount < length \* rotationsPerCm) {      if (digitalRead(IR\_SENSOR\_PIN) == HIGH) {        rotationCount++;        delay(50); // Debounce delay      }    }    stopDCMotor();    Serial.println("Wire feeding complete.");  }  // Cut wire using servo  void cutWire() {    Serial.println("Cutting the wire...");    servoCutter.write(90); // Move cutter servo to cutting position    delay(1000);           // Simulate cutting time    servoCutter.write(0);  // Return servo to initial position    Serial.println("Wire cutting complete.");  }  // Remove sheath using servo  void removeSheath(int length) {    Serial.print("Removing sheath: ");    Serial.print(length);    Serial.println(" cm...");    servoSheath.write(90); // Move servo to sheathing removal position    delay(length \* 500);   // Simulate removal time    servoSheath.write(0);  // Return to initial position    Serial.println("Sheath removal complete.");  }  // Strip and tin cores using soldering servo  void stripAndTinCores() {    Serial.println("Stripping and tinning cores...");    servoSolder.write(90); // Move soldering servo to position    delay(2000);           // Simulate tinning process    servoSolder.write(0);  // Return to initial position    Serial.println("Stripping and tinning complete.");  }  // Stop the DC motor  void stopDCMotor() {    digitalWrite(DC\_MOTOR\_IN1, LOW);    digitalWrite(DC\_MOTOR\_IN2, LOW);    analogWrite(DC\_MOTOR\_ENA, 0);  }  // Display current status on Serial Monitor  void displayStatus() {    Serial.println("=======================");    Serial.print("Wire Presence: ");    Serial.println(wirePresent ? "Detected" : "Not Detected");    Serial.print("Rotation Count: ");    Serial.println(rotationCount);    Serial.print("Error Status: ");    Serial.println(errorFlag ? "ERROR" : "No Errors");    Serial.println("=======================");  }  // Handle errors and halt operation  void handleError(const char\* errorMessage) {    errorFlag = true;    digitalWrite(ERROR\_LED, HIGH);    //digitalWrite(ERROR\_CONTACT\_PIN, HIGH); // Indicate error via dry contact    stopDCMotor();    Serial.print("ERROR: ");    Serial.println(errorMessage);    delay(1000); // Wait before retry  }  // Clear error state  void clearError() {    errorFlag = false;    digitalWrite(ERROR\_LED, LOW);    //digitalWrite(ERROR\_CONTACT\_PIN, LOW); // Reset dry contact  }  // // Web server root handler  // void handleRoot() {  //   server.send(200, "text/html", "<h1>Wire Preparation Machine</h1><p>Use /set?length=VALUE&sheath=VALUE to set parameters.</p>");  // }  // Web server parameter handler  // void handleSetParameters() {    // if (server.hasArg("length")) {      // wireLength = server.arg("length").toInt();    // }    // if (server.hasArg("sheath")) {      // sheathLength = server.arg("sheath").toInt();    // }  //   String message = "<h1>Parameters Updated</h1><p>Wire Length: ";  //   message += wireLength;  //   message += " cm<br>Sheath Length: ";  //   message += sheathLength;  //   message += " cm</p>";  //   server.send(200, "text/html", message);  // } |

The commented parts will be set up properly in real time so that it work properly as required.

## **9. Conclusion**

The wire-prepping machine design meets all the defined objectives, including wire feeding, cutting, sheathing removal, core stripping, and tinning. The design is automated, and error-handling mechanisms are incorporated for reliable operation. The system will be controlled via a local web portal, making it easy to operate and monitor.

**Simulation link:**  
<https://drive.google.com/file/d/1CkSaYsOIHAxI7CTLz5GeMTcRr7Sm1Ert/view?usp=sharing>

**Diagram of the design in proteus:**

