

Mach4 Arduino Modbus TCP/IP

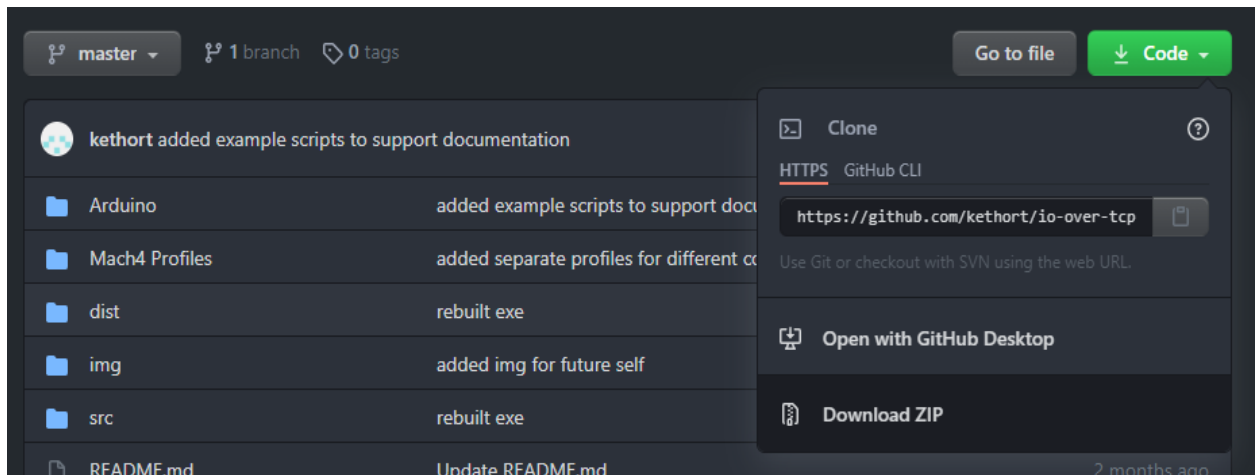
Contents

| | |
|------------------------------------------------------------------------------------------|----|
| Software: | 2 |
| Hardware: | 4 |
| Initial Modbus Connection Setup : | 5 |
| Activate Output Example (Write Coil): | 7 |
| Monitor Inputs Example (Read Coils): | 9 |
| Read Register Example (Read Holding Registers 16bit): | 12 |
| Write Single Register Example (Write Single Register 16bit): | 14 |
| Write a Decimal Value using Two Registers (Write Multiple Registers 16bit): | 16 |
| LUA Mach4 Read/Write Access of Modbus Holding Registers: | 20 |

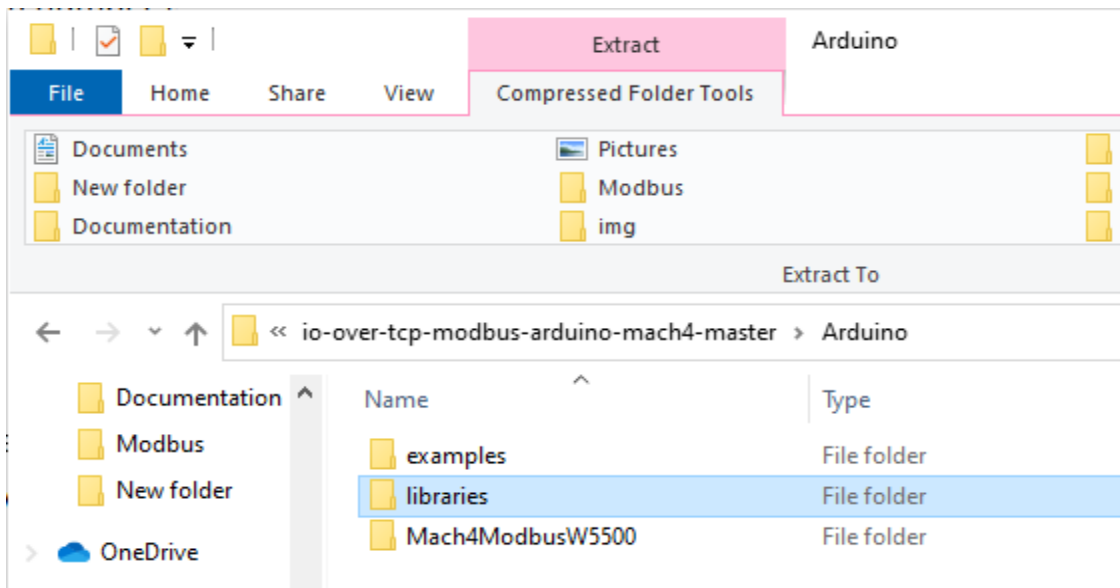
**THIS DOCUMENT ASSUMES YOU ALREADY KNOW HOW TO
UPLOAD SKETCHES TO AN ARDUINO USING THE ARDUINO IDE.
IF YOU DON'T KNOW HOW TO DO THIS, THERE ARE MANY
TUTORIALS ONLINE.**

Software:

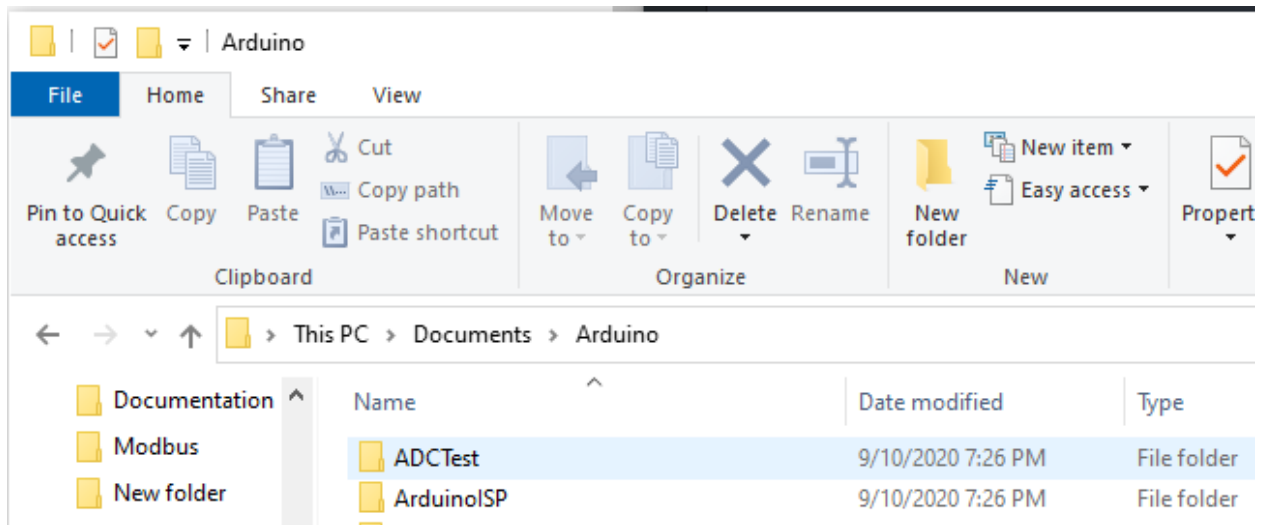
1. Install the latest Arduino IDE – ****NOTE- the instructions in this manual have only been tested using the desktop version of the Arduino software.** (<https://www.arduino.cc/en/software>)
2. Go to this GitHub page <https://github.com/kethort/io-over-tcp-modbus-arduino-mach4>
 - a. Press the Green Code button and select Download zip option.



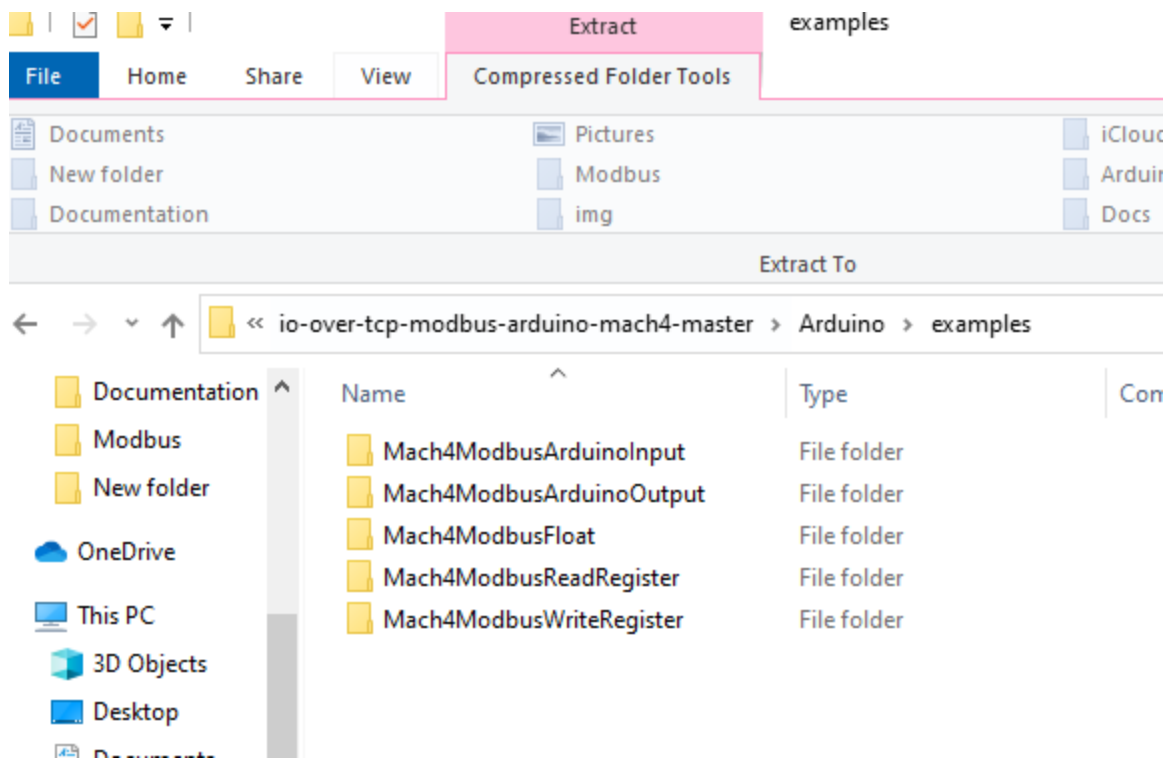
3. Open the zip file and navigate to the Arduino folder.



- Copy the libraries folder to your PC's Documents/Arduino folder









- Open the examples folder from the zip file and copy all the examples over to the Documents/Arduino folder.



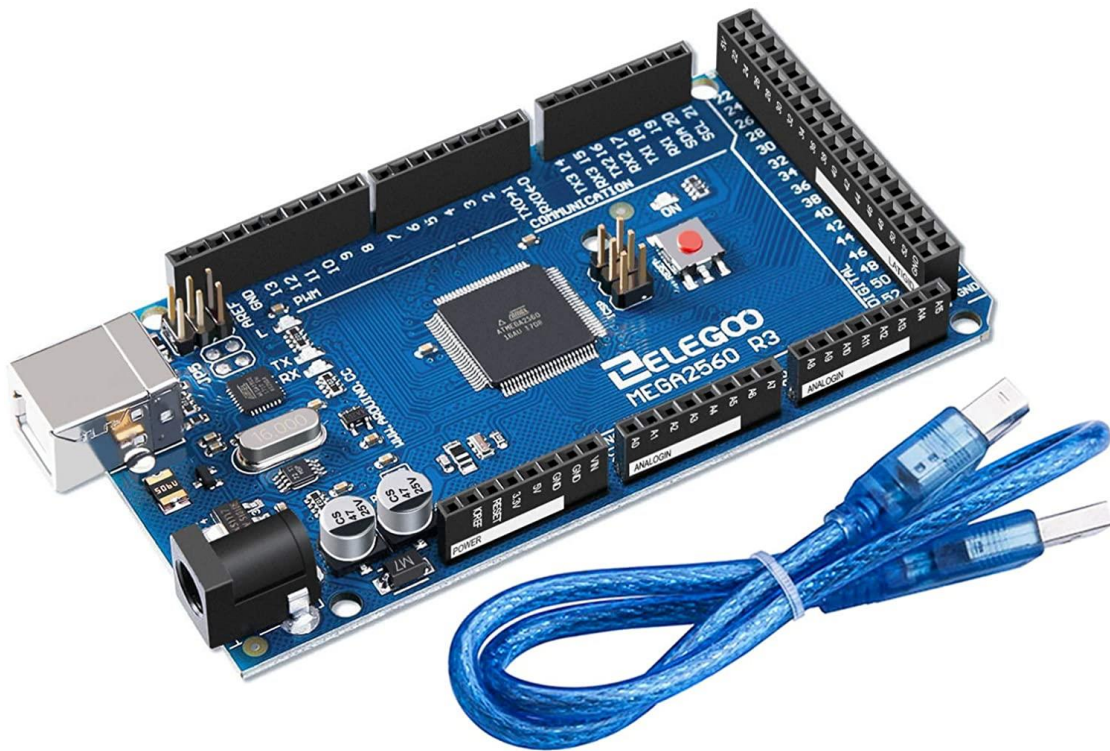
6. In the Mach4 file menu, navigate to Configure->Control->Plugins (tab) and check the mark to enable the LUA, Modbus and Regfile plugins. (this may require a Mach4 restart)

Control Configuration

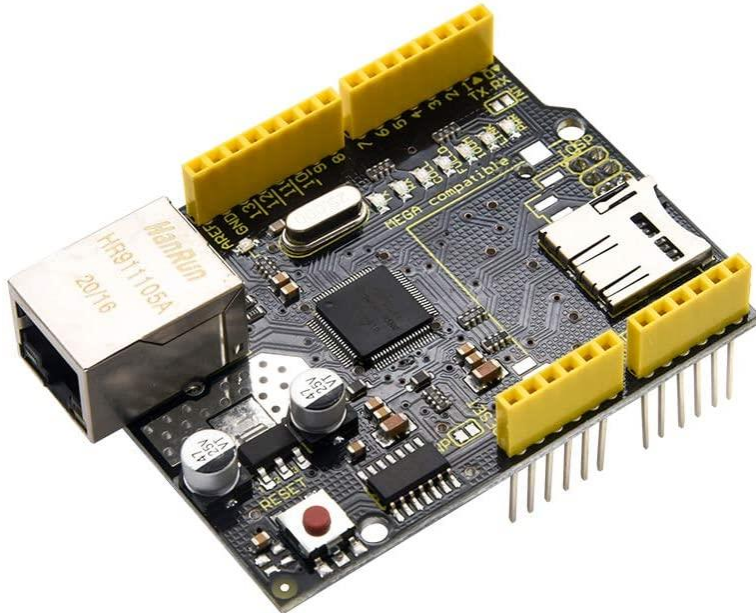
| Defaults | General | Plugins | Motors | Aux. Positions | Axis Mapping | Hor |
|----------|-----------------------------------------------------------------------------------|----------------------------------------|------------|----------------|--------------|-----|
| | Enabled | Description | Version | | | |
| 1 |  | Core - Newfangled Solutions | 4.2.0.4612 | | | |
| 2 |  | Keyboard Inputs - Newfangled Solutions | 4.2.0.4612 | | | |
| 3 |  | LUA - Newfangled Solutions | 4.2.0.4612 | | | |
| 4 |  | Modbus - Newfangled Solutions | 4.2.0.4612 | | | |
| 5 |  | Regfile - Newfangled Solutions | 4.2.0.4612 | | | |
| 6 |  | Serial - Newfangled Solutions | 4.2.0.4612 | | | |

Hardware:

1. Arduino UNO or MEGA. The cheap knock offs work fine.



2. W5500 Ethernet Shield for Arduino.



3. You might also want some LED's, resistors and jumper wires for testing.

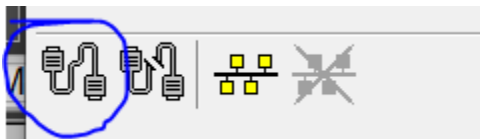
Initial Modbus Connection Setup :

YOU WILL HAVE TO CHANGE THE IP ADDRESS IN EACH OF THE EXAMPLE SKETCHES YOU WANT TO USE TO MATCH YOUR MACH4 MODBUS CONNECTION. BUT THE MODBUS CONNECTION TO THE ARDUINO ONLY HAS TO BE SETUP ONE TIME IN MACH4. ONE MACH4 MODBUS CONNECTION CAN HAVE MULTIPLE DIFFERENT FUNCTIONS INSIDE OF IT.

1. Open one of the example Arduino sketch files that were downloaded from GitHub.
2. Change the IP address in the sketch to match your modbus connection network configuration.

```
4 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};  
5 byte ip[] = {192, 168, 0, 50};
```

3. In the Mach4 file menu, navigate to Configure->Plugins->Modbus.
4. Create a new modbus connection



5. Settings for page 1 in new modbus connection

Add Modbus Connection

Welcome to the Modbus Connection Setup Wizard.

New Connection

Name:

Description:

Modbus Connection Type

☐ Serial ASCII

☐ Serial RTU

☒ TCP

Connection Options

Poll Interval (ms):

Retry Count:

Timeout (ms):

Initial State:

☐ Daniel/Enron 32bit mode.

☒ Swap words on 32 bit integers.

☐ Swap words on Float types.

☒ Use zero based register addressing.

< Back Next > Cancel

6. Settings for page 2 in new modbus connection. Enter the same IP address that was used in the Arduino sketch.

```
Mach4ModbusFloat
1 #include <ModbusIP.h>
2 ModbusIP mb; // create modbus de
3
4 byte mac[] = {0xDE, 0xAD, 0xBE,
5 byte ip[] = {192, 168, 0, 50};
6
7 void setup() {
8   Serial.begin(115200);
9
10  mb.config(mac, ip); // modbus
11
```

Add Modbus Connection

Enter the TCP connection settings.

TCP Settings

IP Address:

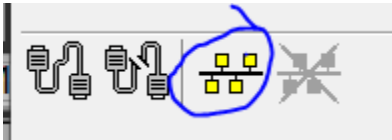
TCP Port:

Activate Output Example (Write Coil):

1. Open the Mach4ModbusArduinoOutput.ino Arduino sketch file.
2. Change the IP address in the sketch to match your modbus connection network configuration.

```
4 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};  
5 byte ip[] = {192, 168, 0, 50};
```

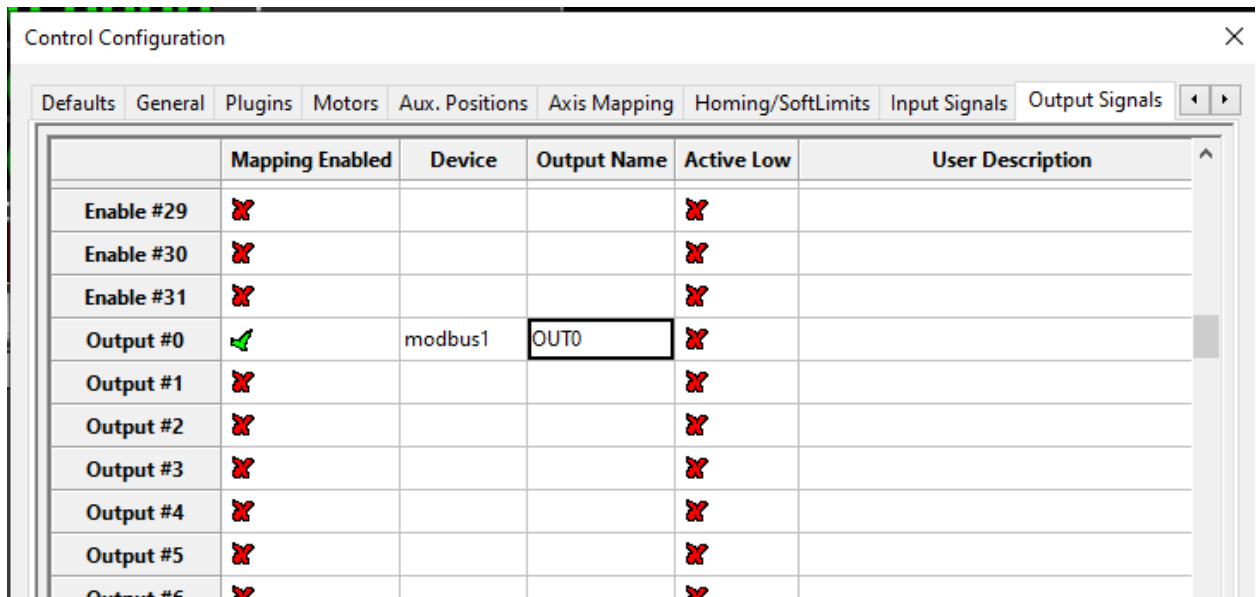
3. Upload the sketch to the Arduino.
4. In the Mach4 file menu, navigate to Configure->Plugins->Modbus.
5. Select the Arduino Modbus Connection and click create function button.



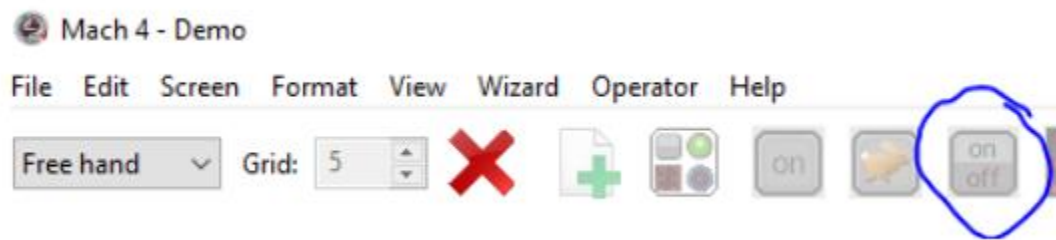
6. Create a write coil function with the following settings. ****NOTE: The modbus register value in the function is 5. In the Arduino sketch, the register used for the modbus coil is also 5.**

A screenshot of the 'Add Modbus Function' dialog box. The title bar says 'Add Modbus Function'. Inside, there's a 'Welcome to the Modbus Function Setup Wizard.' message. Below that, 'Select the Modbus function type:' is followed by a dropdown menu set to 'Write Coil (0x5)'. The configuration fields are: 'Function Name' (writeOutput), 'Slave Address' (1), 'Modbus Register' (5), 'Register Count' (1), 'Register Name Prefix' (OUT), 'Initial State' (Started), 'Scan Multiplier' (1), 'Read As' (Signed), and 'Use I/O?' (unchecked). At the bottom are buttons for 'Help', '< Back', 'Next >', and 'Cancel'. There is also a small image of a coffee cup on the left side of the dialog.

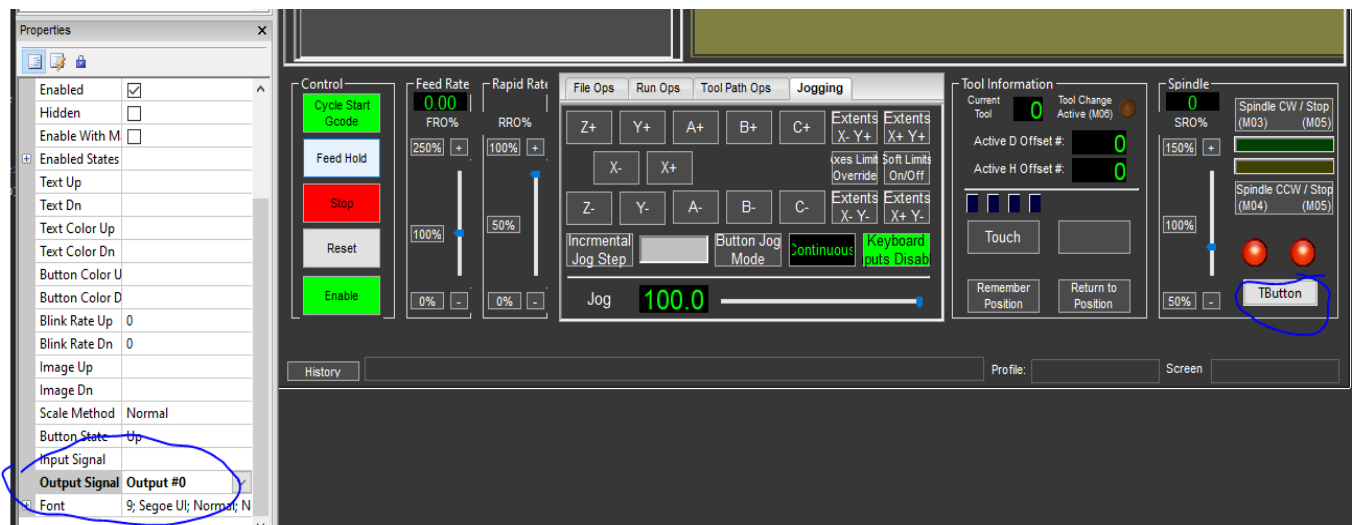
7. Press Next, Finish and Close the Modbus configurator.
8. In the Mach4 file menu, navigate to Configure->Control->Output Signals (tab).
9. Create a new output using the modbus coil function just created.



10. Apply the changes and close the control configurator.
11. In the Mach4 file menu, navigate to Operator->Edit Screen to enter the screen editor.
12. Add a new Toggle Button to the screen.

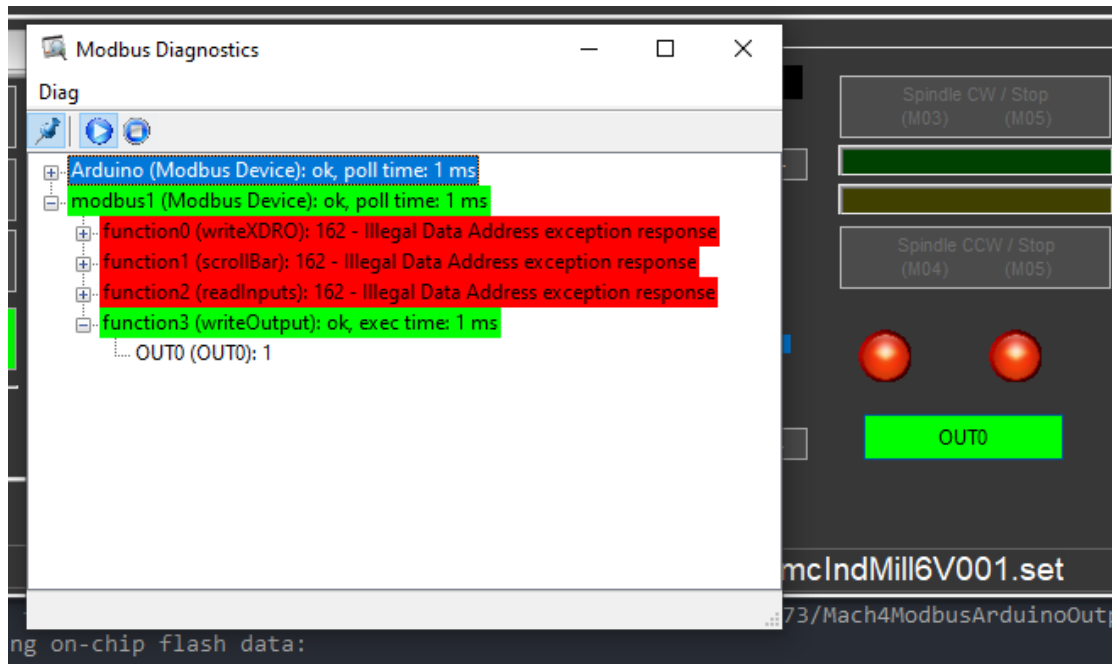


13. Select the Toggle button and edit the properties to map the output signal to the button.



14. In the Mach4 file menu, navigate to Operator->Edit Screen to exit the screen editor.
15. In the Mach4 file menu, navigate to Diagnostic->Modbus and expand the modbus connection to view the state of the write coil register.

- When you press the Toggle button, it should change the state of this register. If the modbus connection is not working, try pressing the stop and play button to refresh the connection.

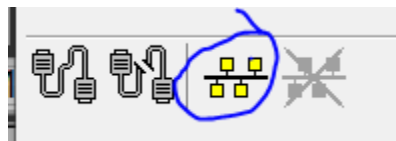


Monitor Inputs Example (Read Coils):

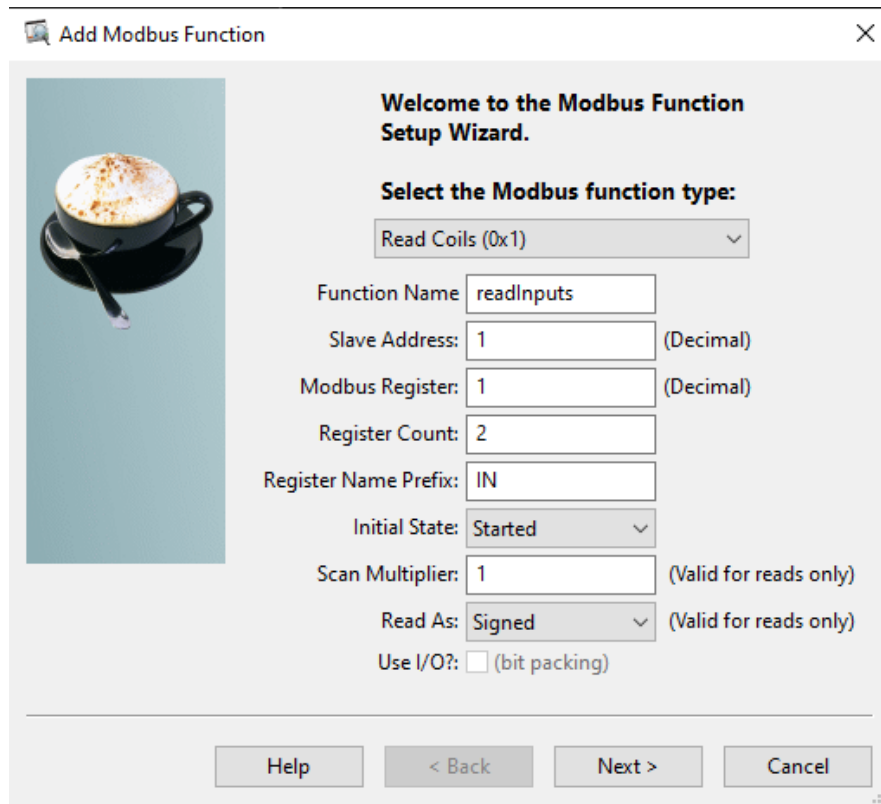
- Open the Mach4ModbusArduinoInput.ino Arduino sketch file.
- Change the IP address in the sketch to match your modbus connection network configuration.

```
4 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};  
5 byte ip[] = {192, 168, 0, 50};
```

- Upload the sketch to the Arduino.
- In the Mach4 file menu, navigate to Configure->Plugins->Modbus.
- Select the Arduino Modbus Connection and click create function button.



6. Create a read coils function with the following settings. ****NOTE: The modbus register value is 1 and the register count is two. This will create two registers 1 and 2. In the Arduino sketch, the registers used for the modbus coils are 1 and 2.**



Add Modbus Function

Welcome to the Modbus Function Setup Wizard.

Select the Modbus function type:

Read Coils (0x1) ▼

Function Name: readInputs

Slave Address: 1 (Decimal)

Modbus Register: 1 (Decimal)

Register Count: 2

Register Name Prefix: IN

Initial State: Started ▼

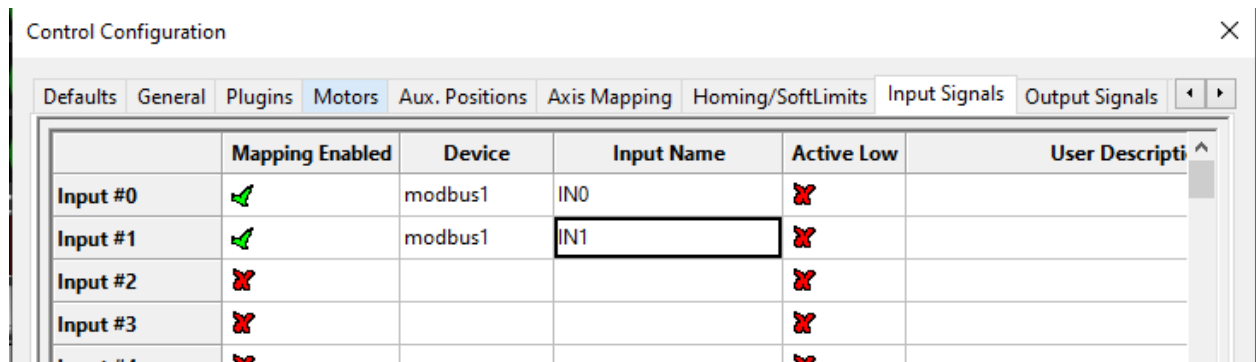
Scan Multiplier: 1 (Valid for reads only)

Read As: Signed ▼ (Valid for reads only)

Use I/O?: ☐ (bit packing)

Help < Back Next > Cancel

7. Press Next, Finish and Close the Modbus configurator.
8. In the Mach4 file menu, navigate to Configure->Control->Input Signals (tab).
9. Create two new inputs using the modbus coil function just created.



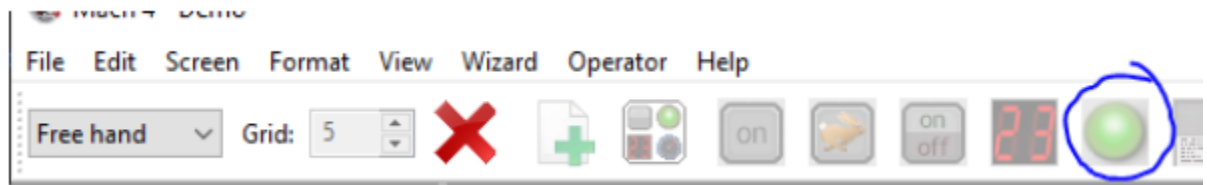
Control Configuration

Defaults General Plugins Motors Aux. Positions Axis Mapping Homing/SoftLimits Input Signals Output Signals

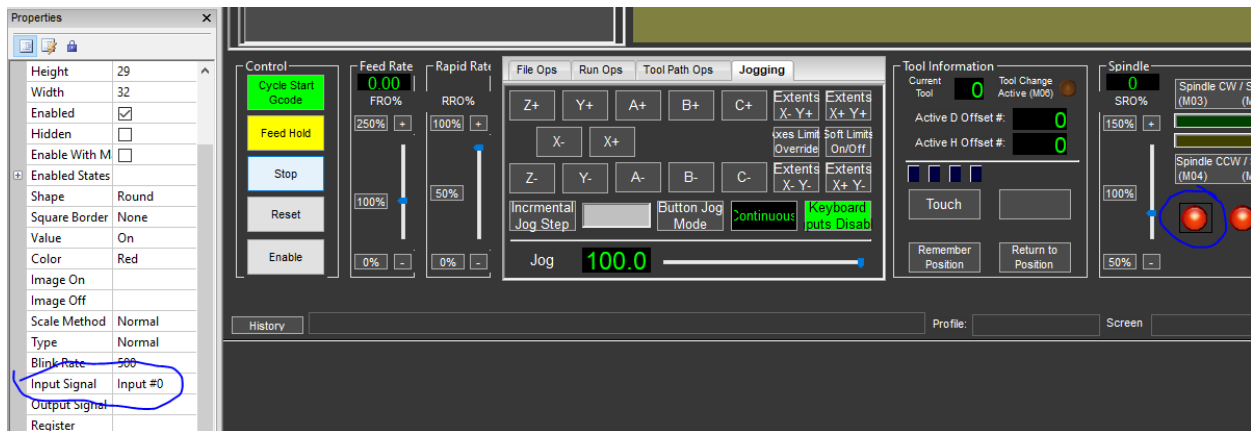
| | Mapping Enabled | Device | Input Name | Active Low | User Descripti |
|----------|-----------------|---------|------------|------------|----------------|
| Input #0 | ✔ | modbus1 | IN0 | ✗ | |
| Input #1 | ✔ | modbus1 | IN1 | ✗ | |
| Input #2 | ✗ | | | ✗ | |
| Input #3 | ✗ | | | ✗ | |
| Input #4 | ✗ | | | ✗ | |

10. Apply the changes and close the control configurator.
11. In the Mach4 file menu, navigate to Operator->Edit Screen to enter the screen editor.

12. Add two new LED's to the screen.



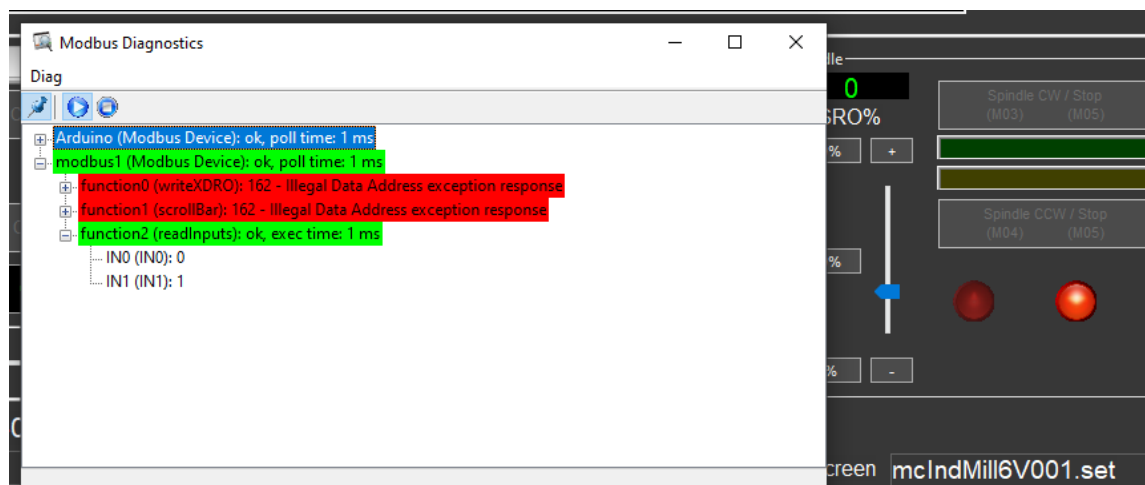
13. Select each LED and edit the properties to map the input signal to the LED.



14. In the Mach4 file menu, navigate to Operator->Edit Screen to exit the screen editor.

15. In the Mach4 file menu, navigate to Diagnostic->Modbus and expand the modbus connection to view the state of the read coil function.

16. When the input on the Arduino is triggered, the LED should turn on or off and the state in the diagnostics window will change. If the modbus connection is not working try pressing the stop and play button to refresh the connection.

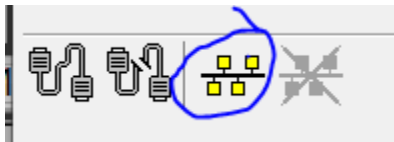


Read Register Example (Read Holding Registers 16bit):

1. Open the Mach4ModbusReadRegister.ino Arduino sketch file.
2. Change the IP address in the sketch to match your modbus connection network configuration.

```
4 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};  
5 byte ip[] = {192, 168, 0, 50};
```

3. Upload the sketch to the Arduino.
4. In the Mach4 file menu, navigate to Configure->Plugins->Modbus.
5. Select the Arduino Modbus Connection and click create function button.



6. Create a read holding registers 16bit function with the following settings. ****NOTE: The modbus register value in the function is 87. In the Arduino sketch, the register used for the modbus register is also 87.**

Add Modbus Function [X]

Welcome to the Modbus Function Setup Wizard.

Select the Modbus function type:
Read Holding Registers 16bit (0x3) [v]

Function Name: readRandom

Slave Address: 1 (Decimal)

Modbus Register: 87 (Decimal)

Register Count: 1

Register Name Prefix: rand

Initial State: Started [v]

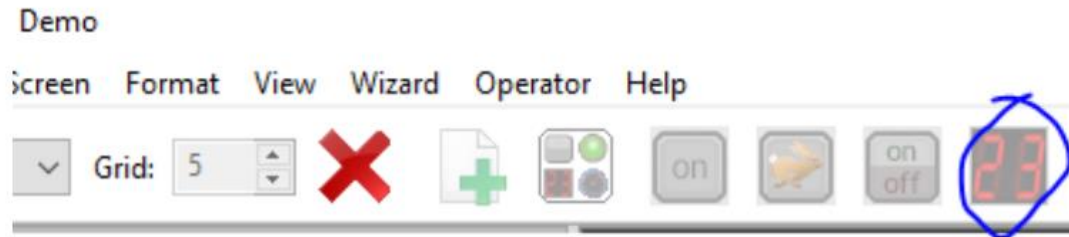
Scan Multiplier: 1 (Valid for reads only)

Read As: Signed [v] (Valid for reads only)

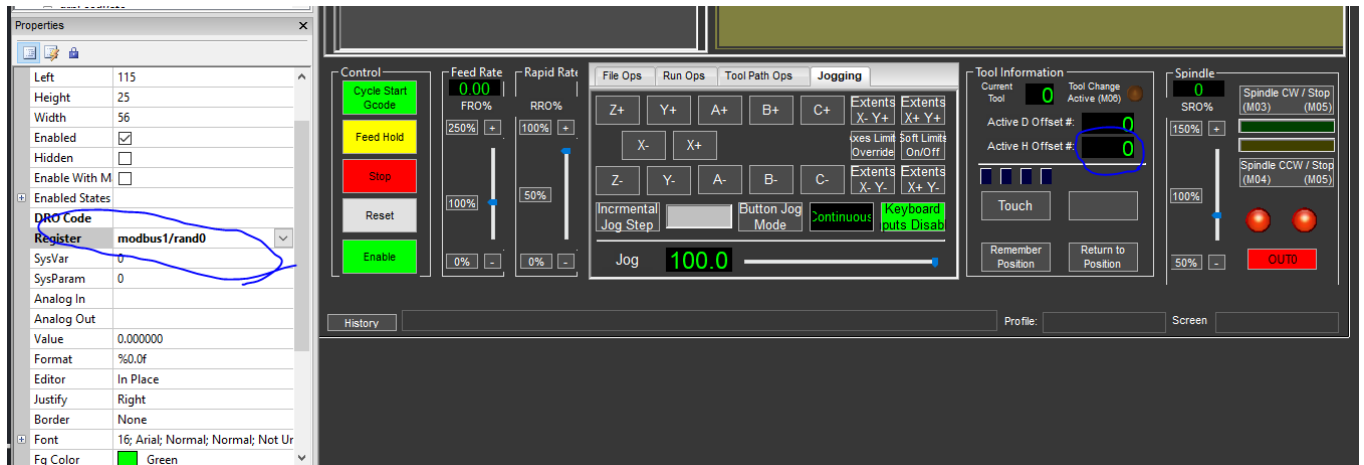
Use I/O?: ☐ (bit packing)

[Help] [< Back] [Next >] [Cancel]

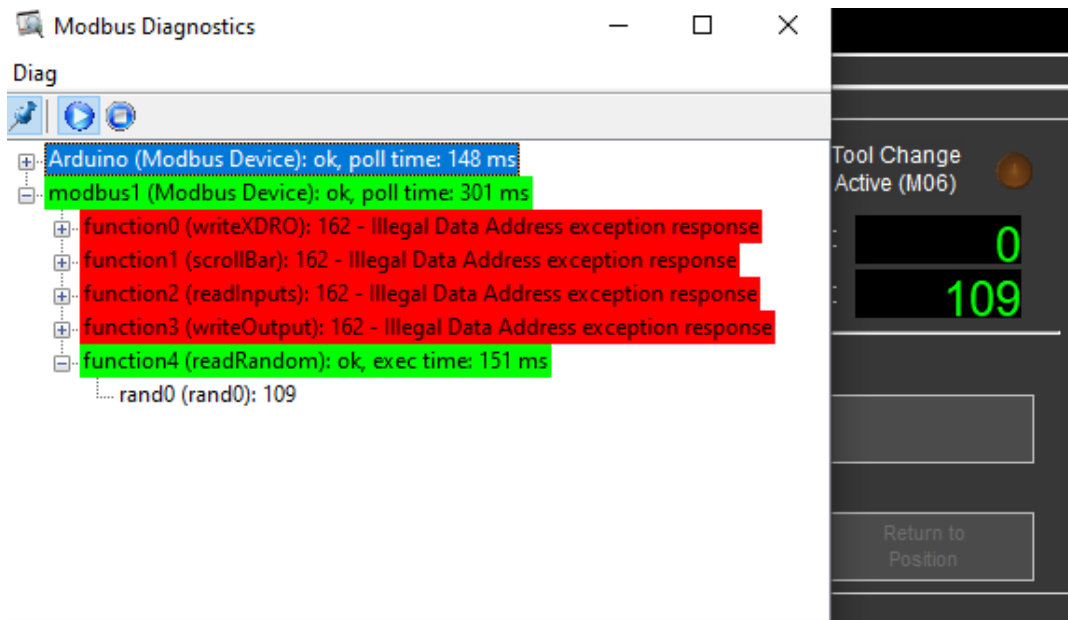
7. Press Next, Finish and Close the Modbus configurator.
8. In the Mach4 file menu, navigate to Operator->Edit Screen to enter the screen editor.
9. Add a new DRO to the screen.



10. Select the DRO and edit the properties to map the register to the DRO.



11. In the Mach4 file menu, navigate to Operator->Edit Screen to exit the screen editor.
12. In the Mach4 file menu, navigate to Diagnostic->Modbus and expand the modbus connection to view the state of the read holding register function.
13. The DRO should update every 150 milliseconds with random numbers between 0 and 300.

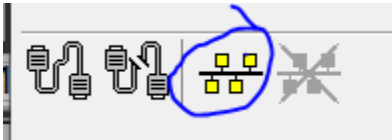


Write Single Register Example (Write Single Register 16bit):

1. Open the Mach4ModbusWriteRegister.ino Arduino sketch file.
2. Change the IP address in the sketch to match your modbus connection network configuration.

```
4 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};  
5 byte ip[] = {192, 168, 0, 50};
```

3. Upload the sketch to the Arduino.
4. In the Mach4 file menu, navigate to Configure->Plugins->Modbus.
5. Select the Arduino Modbus Connection and click create function button.



6. Create a write single register function with the following settings. ****NOTE: The modbus register value is 58. If you look at the Arduino sketch, the register used for the modbus hreg is 58.**

A screenshot of the 'Add Modbus Function' dialog box in Mach4. The dialog has a title bar with a close button. On the left is a decorative image of a coffee cup. The main area is titled 'Welcome to the Modbus Function Setup Wizard.' and 'Select the Modbus function type:'. A dropdown menu shows 'Write Single Register 16bit (0x6)'. Below this are several input fields: 'Function Name' (scrollBar), 'Slave Address' (1), 'Modbus Register' (58), 'Register Count' (1), 'Register Name Prefix' (scrollBar), 'Initial State' (Started), 'Scan Multiplier' (1), 'Read As' (Signed), and 'Use I/O?' (unchecked). At the bottom are buttons for 'Help', '< Back', 'Next >', and 'Cancel'.

Add Modbus Function

Welcome to the Modbus Function Setup Wizard.

Select the Modbus function type:

Write Single Register 16bit (0x6)

Function Name: scrollBar

Slave Address: 1 (Decimal)

Modbus Register: 58 (Decimal)

Register Count: 1

Register Name Prefix: scrollBar

Initial State: Started

Scan Multiplier: 1 (Valid for reads only)

Read As: Signed (Valid for reads only)

Use I/O?: ☐ (bit packing)

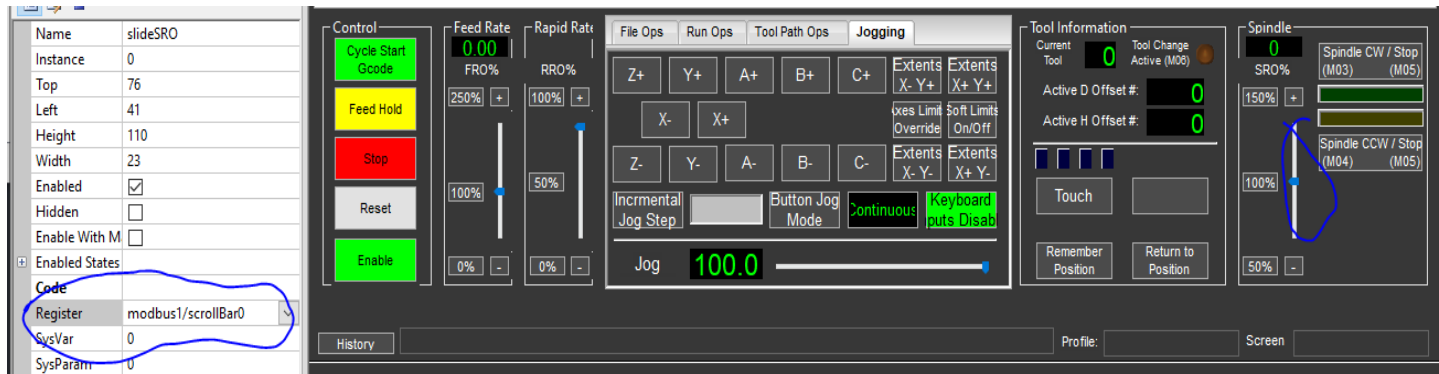
Help < Back Next > Cancel

7. Press Next, Finish and Close the Modbus configurator.
8. In the Mach4 file menu, navigate to Operator->Edit Screen to enter the screen editor.

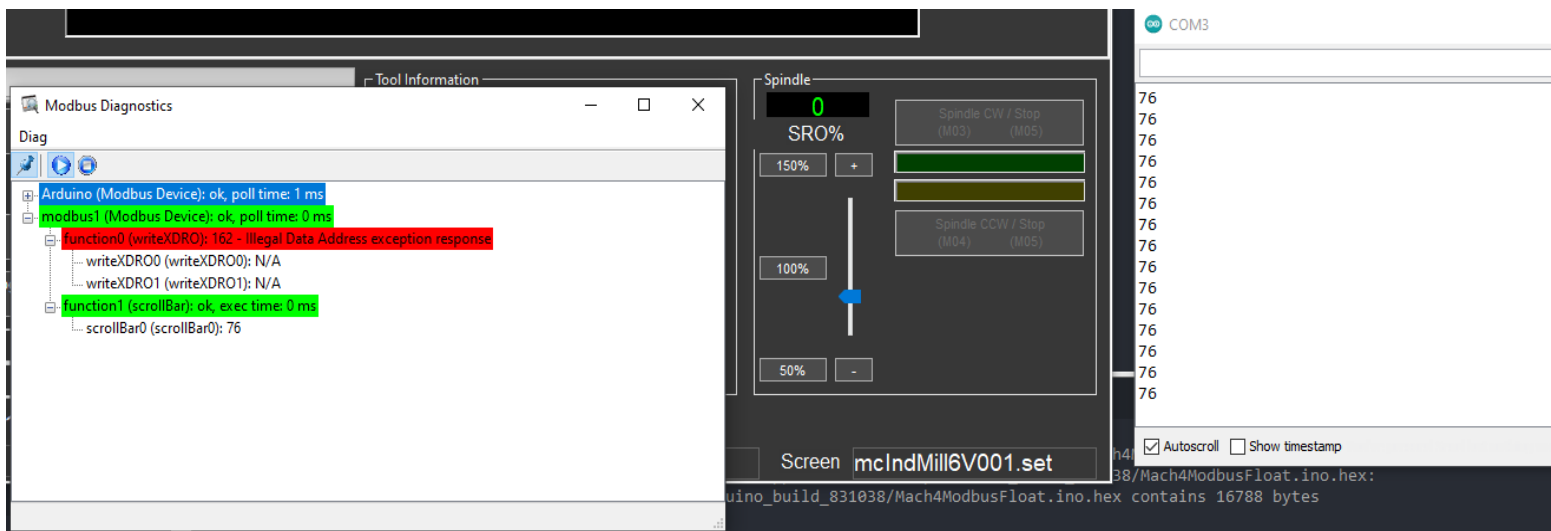
9. Add a slider bar to the screen.



10. Select the slider bar and edit the properties to map the register to the slider bar.



11. In the Mach4 file menu, navigate to Operator->Edit Screen to exit the screen editor.
12. In the Mach4 file menu, navigate to Diagnostic->Modbus and expand the modbus connection to view the state of the write register function.
13. The value of the slider bar can be monitored using the serial monitor in the Arduino IDE.
- **NOTE: when the serial monitor is opened in the Arduino IDE, the Arduino is reset, so the modbus communication will restart and needs to be restarted in Mach4 by pressing the Stop then Play button in the Modbus Diagnostics window.**
14. When the slider bar is changed in Mach4, the value printed in the Arduino serial monitor should reflect the value of the slider bar.

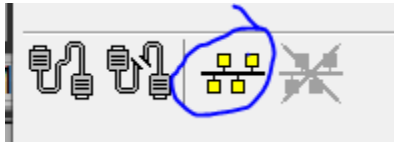


Write a Decimal Value using Two Registers (Write Multiple Registers 16bit):

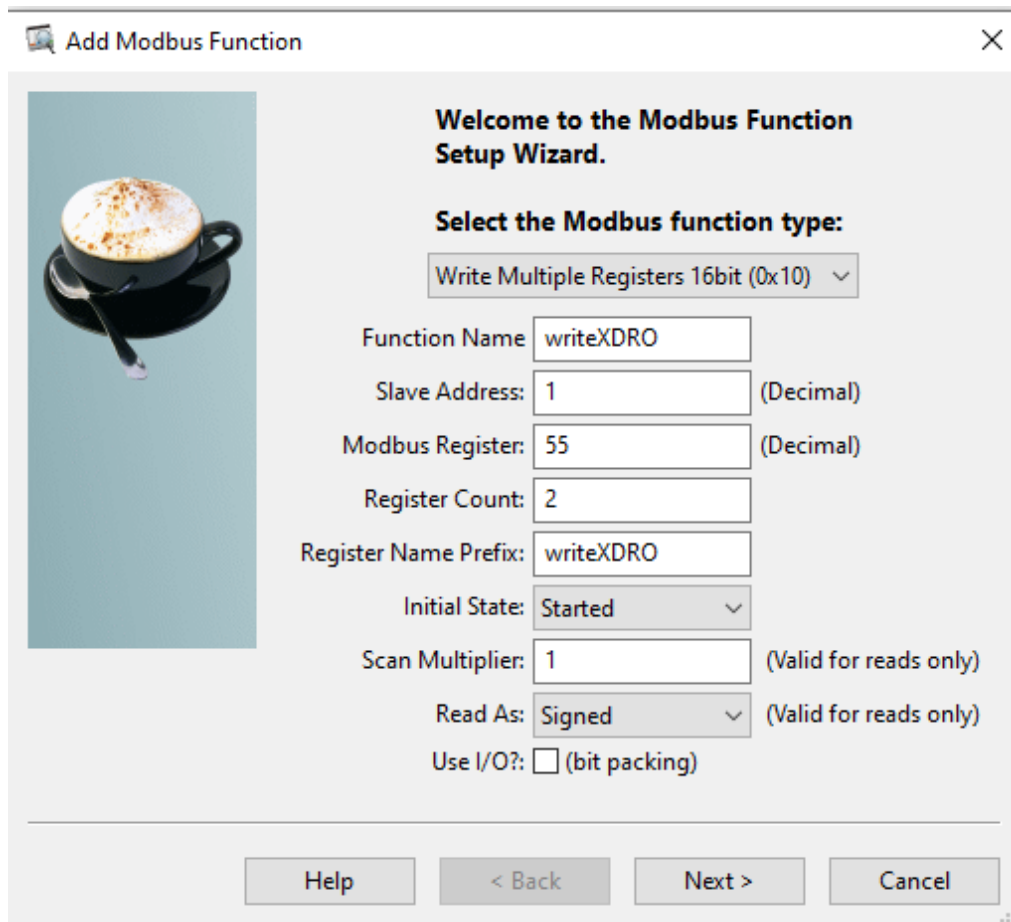
1. Open the Mach4ModbusFloat.ino Arduino sketch file.
2. Change the IP address in the sketch to match your modbus connection network configuration.

```
3  
4 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};  
5 byte ip[] = {192, 168, 0, 50};
```

3. Upload the sketch to the Arduino.
4. In the Mach4 file menu, navigate to Configure->Plugins->Modbus.
5. Select the Arduino Modbus Connection and click create function button.



6. Create a write multiple registers function with the following settings. ****NOTE: The modbus register value starts at 55 and has a count of 2. If you look at the Arduino sketch, the registers used for the modbus hregs are 55 and 58.**



Add Modbus Function

Welcome to the Modbus Function Setup Wizard.

Select the Modbus function type:

Write Multiple Registers 16bit (0x10) ▾

Function Name: writeXDRO

Slave Address: 1 (Decimal)

Modbus Register: 55 (Decimal)

Register Count: 2

Register Name Prefix: writeXDRO

Initial State: Started ▾

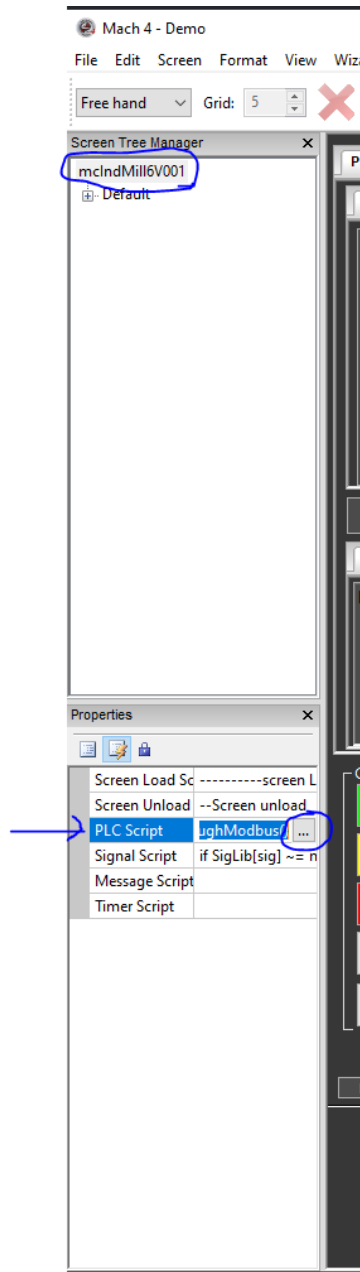
Scan Multiplier: 1 (Valid for reads only)

Read As: Signed ▾ (Valid for reads only)

Use I/O?: ☐ (bit packing)

Help < Back Next > Cancel

7. Press Next, Finish and Close the Modbus configurator.
8. In the Mach4 file menu, navigate to Operator->Edit Screen to enter the screen editor.
9. Highlight the screen name in the Screen Tree Manager and edit the PLC script by pressing the Ellipses button next to the PLC Script section in the Properties.



10. Copy and paste the code from the PLC.lua file (located in the examples directory of the Github zip file) into the PLC script. This code monitors the X axis position and sends the position to the Arduino via modbus. ***NOTE: be careful not to overwrite or delete anything in the PLC script, unless you know what you are doing.*

```

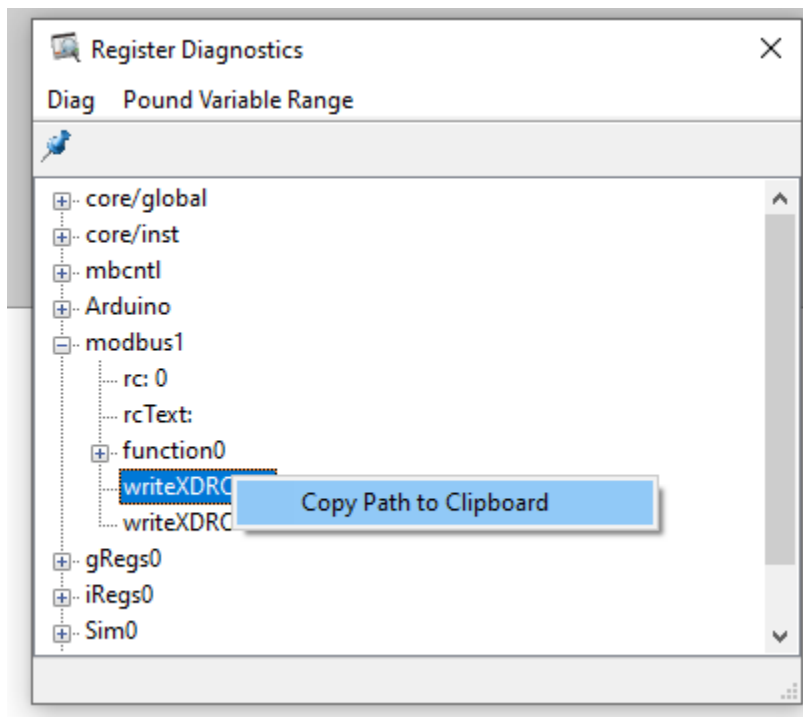
1  local inst = mc.mcGetInstance()
2  local pc = 0;
3  testcount = testcount + 1
4  machState, rc = mc.mcCntlGetState(inst);
5  local inCycle = mc.mcCntlIsInCycle(inst);
6
7  function sendDROFloatThroughModbus()
8      -- get the DRO value from Mach4 and form
9      local axisPos = mc.mcAxisGetPos(inst, mc
10     axisPos = tonumber(string.format('%0.4f'
11
12     -- get the non-decimal part of the x axi
13     local axisPosPrefix = 0
14     if axisPos < 0 then
15         axisPosPrefix = math.ceil(axisPos)
16     else
17         axisPosPrefix = math.floor(axisPos)
18     end
19
20     -- get the decimal part of the DRO value
21     local axisPosDecimal = (axisPos - axisPosPrefix)
22
23     -- get the modbus register handles
24     local xAxisDROReg1 = mc.mcRegGetHandle(
25     local xAxisDROReg2 = mc.mcRegGetHandle(
26
27     -- set the first register value to the r
28     mc.mcRegSetValueLong(xAxisDROReg1, axisPosDecimal)
29     droRegsPos = droRegsPos + 1
30
31     -- set the second register value to the
32     mc.mcRegSetValueLong(xAxisDROReg2, axisPosPrefix)
33 end
34
35 sendDROFloatThroughModbus()
36

```

11. In the Mach4 file menu, navigate to Operator->Edit Screen to exit the screen editor.
12. In the Mach4 file menu, navigate to Diagnostic->Modbus and expand the modbus connection to view the state of the write multiple registers function.
13. The value of the decimal number can be monitored using the serial monitor in the Arduino IDE.
***NOTE: when the serial monitor is opened in the Arduino IDE, the Arduino is reset, so the modbus communication will restart and needs to be restarted in Mach4 by pressing the Stop then Play button in the Modbus Diagnostics window.*
14. Enable the machine and move the X axis. When the X axis moves and the DRO changes, the registers are updated and sent to the Arduino via modbus (image below).

LUA Mach4 Read/Write Access of Modbus Holding Registers:

1. The holding registers are stored in the Regfile and can be read from, or written to, using the Mach LUA API. The path of the register can be accessed by opening the register diagnostic window, right-click the register and select copy path to clipboard.



The register can be read to or written to from (the register path that was copied is placed in the quotes of the mcRegGetHandle function):

```
1  -- get the modbus register handle
2  local xAxisDR0Reg1 = mc.mcRegGetHandle(inst, 'modbus1/writeXDR00')
3
4  -- read the value of the register
5  local xAxisDR0Reg1Value = mc.mcRegGetValue(xAxisDR0Reg1)
6
7  -- write the number 100 to the register
8  mc.mcRegSetValueLong(xAxisDR0Reg1, 100)
```