**Turntable Arduino Sketch v2 by Anthony Kochevar**

**1-2024**

1. **Usage Pre-requisites**

The code provided works with 28BYJ-48 stepper motors and uses the Accelstepper library found here:

https://www.arduino.cc/reference/en/libraries/accelstepper

If you are not using a 28BYJ-48 stepper and basic board you will need to modify the sketch to match your stepper motor you are using. It is also good to know how many steps per revolution your stepper has when using this library, in most cases it will be 2048. If you think your number of steps per revolution might be greater than 2048, set the **stepsPerRevolution** to a higher number for now. More on this later. Make the changes to the code if needed and upload to the Arduino. Open the serial monitor on the Arduino IDE and set it to 19200 baud so you can see the output.

1. **Initial Setup**

Power up the Arduino. Ignore output position in the serial monitor for now but pay attention to clockwise and counterwise direction messages. Move the rotary encoder and watch the turntable. The turntable should move in the direction you moved the encoder, if it is moving the opposite direction, revisit your stepper motor initialization line in step one and reverse the pins the motor is using in the code. To stop the turntable moving press and release the rotary encoder’s button. LED light will be on if the turntable is still moving and off when not.

Manually move the turntable by rotating in both directions and make sure the turntable moves clockwise and counterwise correctly visually and with the output messages in the serial monitor. Again, ignore position output for now.

Move the turntable to where you want your **home (0)** position to be. Use some mark or layout scenery for your home position. You cannot use **home** or **stepPerRevolution** as an indexed position so it is best to have this position pointing to scenery.

Once the turntable is visually in the **home (0)** position you want to use. Reboot the Arduino by unplugging and plugging it back in. Follow directions on the serial monitor to set the home position. You do this by pressing and holding the rotary encoder button as the Arduino boots. You should see a message that home was set and position is now zero. If you want you can press and hold the rotary button for 10 seconds and then release it. This also sets the current position to zero. LED will blink three times to show position has been set to 0.

Once home is set, to get the number of steps per revolution, manually move the turntable clockwise by rotating the encoder clockwise. You can set the speed by moving the encoder more in the same direction and slow it down moving opposite. To stop movement press and release the encoder button. Make sure the turntable makes a complete revolution and reset itself correctly when it gets back to the home (0) position. Check the output in the serial monitor. Make any changes to the **stepsPerRevolution** variable in the code if needed and reupload. Keep doing this until you are satisfied that the steps are resetting correctly when the turntable crosses home.

1. **Getting and Setting Index Positions**

If step 2 was done correctly and working as expected you can move on to get index positions for your tracks. The code is written for 64 possible index positions but with a few changes can handle up to **stepsPerRevolution** – 2. 64 will likely be plenty for most layouts. Starting at or near zero, move the turntable clockwise slowly until it lines up with each of your track offshoots. Make a note of the position output in the serial monitor. Set the encoder to as low as it will go for very slow fine movement and press and release the button to stop.

Get each track position going from lowest to highest moving clockwise. These values must be entered this way in the array for the code to work properly. In the Arduino code enter the total number of indexed positions your turntable will have in the **numPos** variable. Then enter each position value in the **indexPos** array in the code. Again, it needs to go from lowest to highest moving clockwise. Once done reupload to the Arduino. In the serial monitor you should see the position reload correctly from EEPROM each time the Arduino power cycles or reboots. Make sure this is happening correctly.

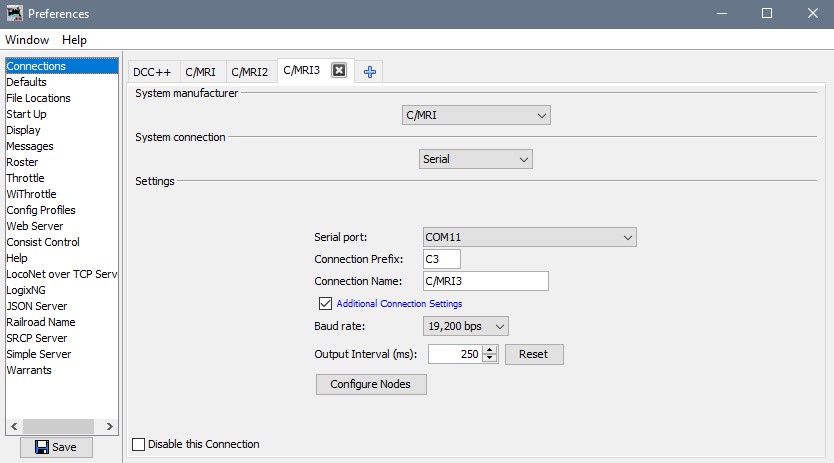
Test an index move by pressing and holding the rotary encoder button for 2 seconds and releasing. You should see the turntable move to the next indexed position you set in your code. Test that it goes to all positions correctly then back to the first.

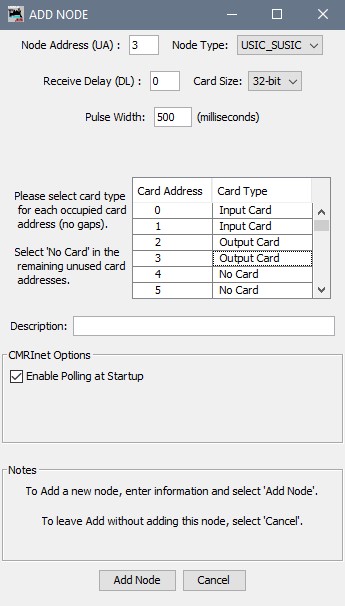
**Note:** If the turntable is in the first or last indexed position or beyond them, the turntable will move in the opposite direction back to the first or last position. Also a index move uses a set speed in the code. Index moves also use acceleration and deceleration when moving the turntable.

1. **JMRI-CMRI Setup**

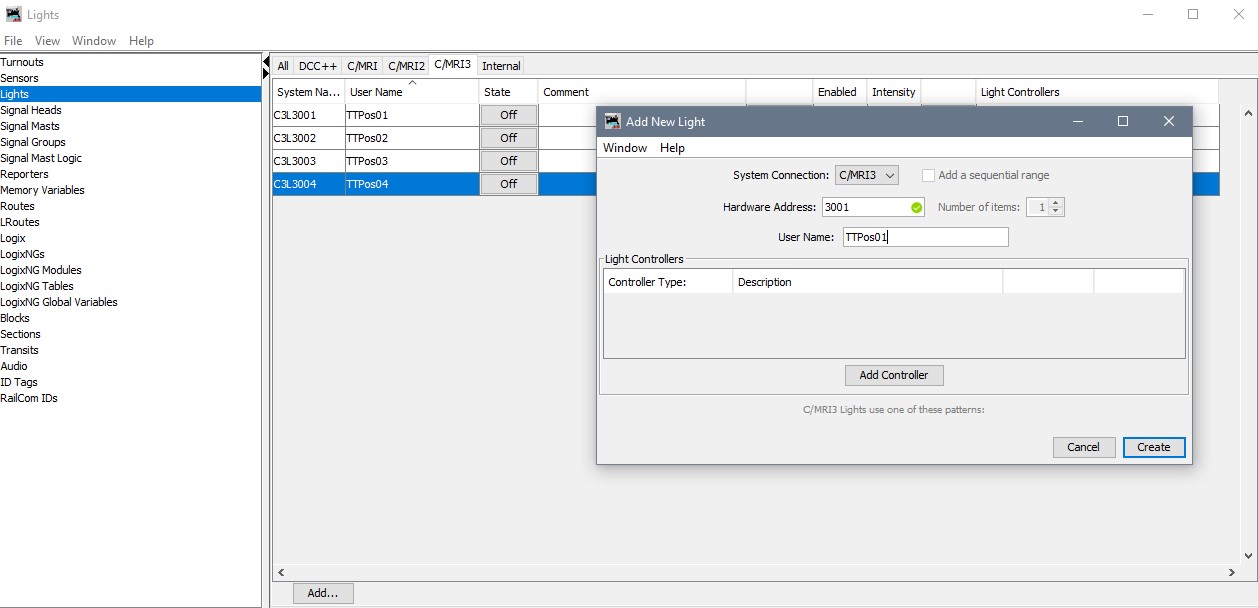
Note: While the manual and index portion using the switch and pots works really well, using the JMRI-CMRI has some bug and issues. I’ll go over these later.

The code comes set with the CMRI node being 3. This means that JMRI will start with address 3001. You can change the node address in the code to be what you want. If you change it to 1 JMRI will start with address 1001. In this example we will use it at three. Create a CMRI connection and nodes as shown in the screenshots below. You’ll need the Com port number your Arduino is connected at as well.



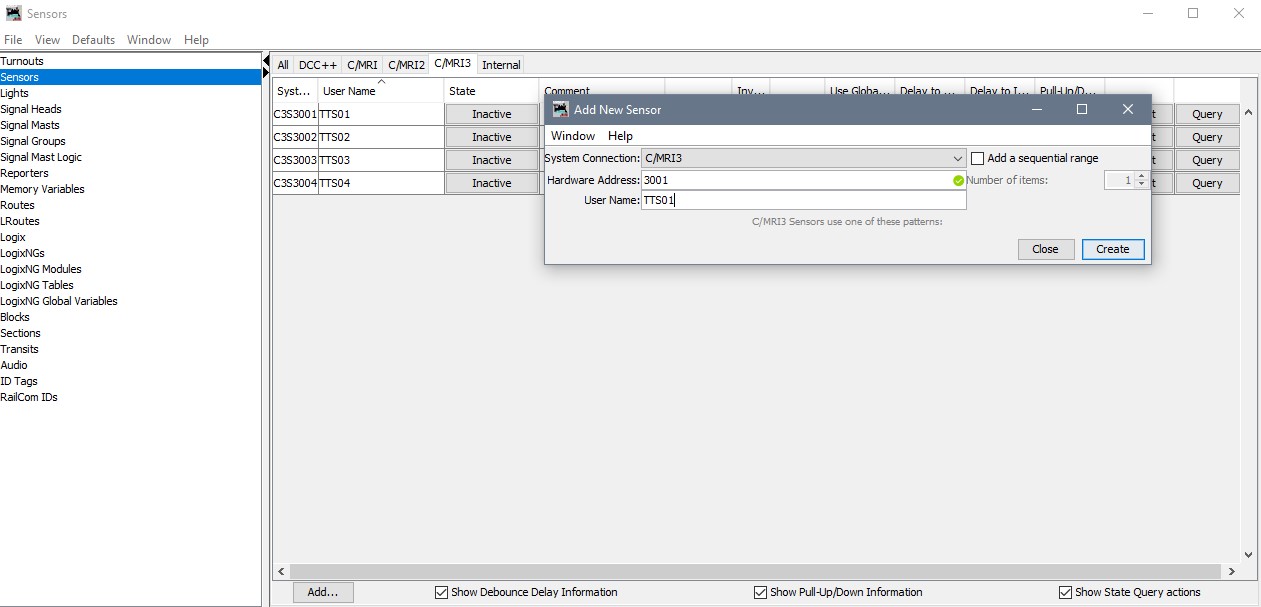


Save the configuration and restart JMRI. Open the lights table and in the correct CMRI node create a light for each index position on your turntable. Again, start with the lowest or first indexed position moving clockwise around the turntable to the last position. In our example we have 4 indexed positions so we will create a light for each starting at address 3001 to 3004.

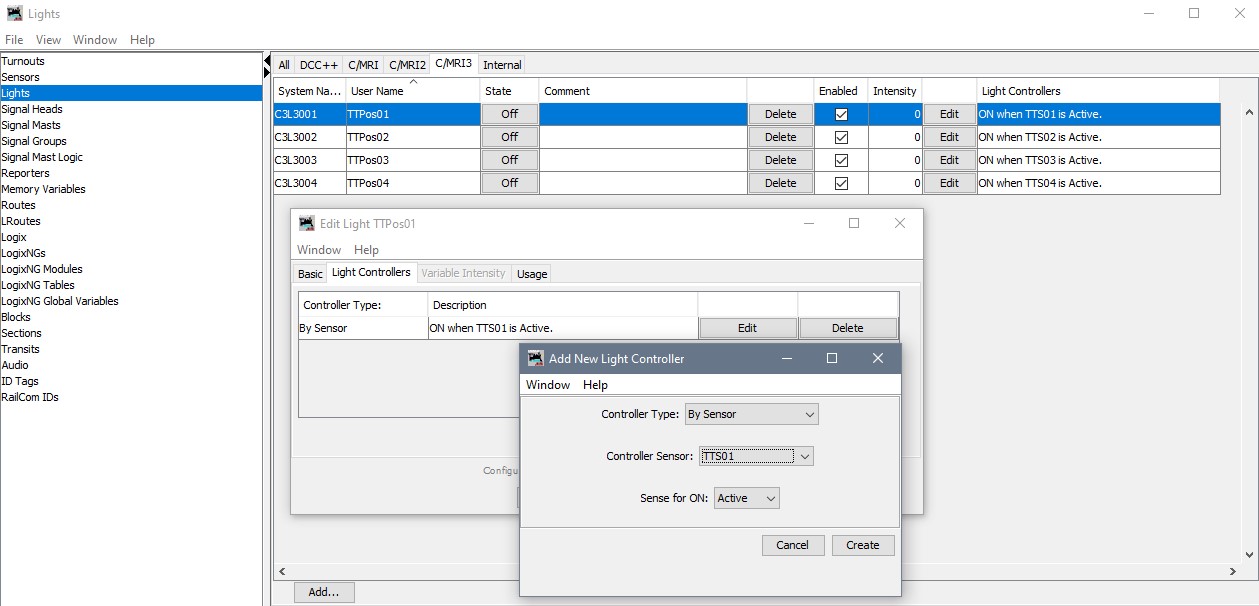


Test that the turntable moves to that position by turning the “light” to On. Turn off that light then turn another light to On and the turntable should move to that position. **Note:** Only one light can be on at a time or the turntable will move back and forth in an infinite loop.

Go to the Sensors table. Create a sensor for each of your index positions. **Note:** You will use the same hardware address as you did for the light as the light is the output side of the address and sensor is the input side of the address.



You’ll notice the sensors will all stay off most of the time. The code in the Arduino does this. Now go back to the lights table. Go through each light and set up a light controller to use the sensor using the same address as the light. Set it to turn the light on when active.



Now go back to the sensors table and turn on a sensor. It will stay active for a short time and your turntable should move to the position the sensor is controlling. The sensors will go to unknown while it is moving. After it stops, they will all go back to inactive. You can now place the sensors on your panels and use them as buttons.

1. **Bugs an Issues**

What would an app be with out bugs! All the bugs and issues center around the JMRI-CMRI side of things. It is possible it is my code but I think it is more a timing or buffer issue with JMRI.

1. The first issue is going back and forth from using JMRI and the switch. If you use JMRI to move the turntable then afterwards try to use the switch (ether a manual move or index move) the turntable will move as expected. However, when it stops it will immediately move back to the last move made by JMRI. Once done though you can use the switch again and it will move and stay where expected.
2. Another issue is when activating a sensor in JMRI, sometimes it doesn’t take and you’ll need to set to inactive and then active again.

It is possible these are due to faulty code but I’m not sure. I think it is more a JMRI or CMRI buffer or timing issue. If it is my code and you find a fix, please contact me and I’ll re-post it. I’ve done a lot of testing with the switch using both a manual and index move and it seems to work as expected always.