Making an odom topic from wheel encoders

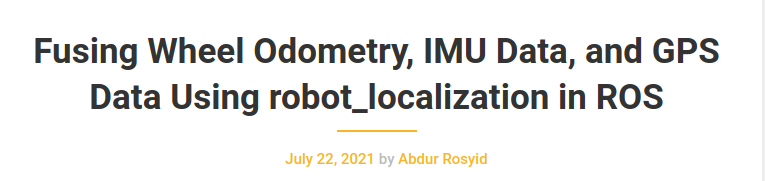
<https://automaticaddison.com/how-to-publish-wheel-encoder-tick-data-using-ros-and-arduino/>

<https://automaticaddison.com/how-to-publish-wheel-odometry-information-over-ros/>

<https://answers.ros.org/question/241602/get-odometry-from-wheels-encoders/>

Broader topic:

<https://blog.abdurrosyid.com/2021/07/22/fusing-wheel-odometry-imu-data-and-gps-data-using-robot_localization-in-ros/>



<https://answers.ros.org/question/361879/generating-empty-map-for-outdoor-navigation/>

Test Arduino access

Get code from Arduino Example

| /\*  \* rosserial Subscriber Example  \* Blinks an LED on callback  \*/  #include <ros.h>  #include <std\_msgs/Empty.h>  ros::NodeHandle nh;  void messageCb( const std\_msgs::Empty& toggle\_msg){  digitalWrite(LED\_BUILTIN, HIGH-digitalRead(LED\_BUILTIN)); // blink the led  }  ros::Subscriber<std\_msgs::Empty> sub("toggle\_led", &messageCb );  void setup()  {  pinMode(LED\_BUILTIN, OUTPUT);  nh.initNode();  nh.subscribe(sub);  }  void loop()  {  nh.spinOnce();  delay(1);  } |
| --- |

Store list of devices connected with power off; $ ls /dev/ > dev\_list\_1.txt

Turn on power to the Arduino Board

Then run this after you plug it in; $ ls /dev/ | diff --suppress-common-lines -y - dev\_list\_1.txt

al@al-HP-ProBook-645-G1:~$ ls /dev/ | diff --suppress-common-lines -y - dev\_list\_1.txt

serial <

ttyACM3 <

al@al-HP-ProBook-645-G1:~$

| byte requestResult;  byte readArray[2];  uint16\_t readValue = 0;  Wire.beginTransmission(AS5048\_ADDRESS);  Wire.write(AS5048B\_ANGLMSB\_REG);  requestResult = Wire.endTransmission(false);  if (requestResult){  Serial.print("I2C error: ");  Serial.println(requestResult);  }  Wire.requestFrom(AS5048\_ADDRESS, 2);  for (byte i=0; i < 2; i++) {  readArray[i] = Wire.read();  }  readValue = (((uint16\_t) readArray[0]) << 6);  readValue += (readArray[1] & 0x3F);  return readValue;  current\_position = AMS\_AS5048B\_readReg16();  //reference: next 5 lines are based on magic code from Jeff Sampson - thank you!  bitshift\_cur\_pos = current\_position << 2; //Bitshiftleft - The leftmost 2 bits in current\_position are shifted out of existence  bitshift\_last\_pos = last\_position << 2;  bitshift\_pos\_delta = (bitshift\_cur\_pos - bitshift\_last\_pos);  position\_movement = bitshift\_pos\_delta >> 2; //BitshiftRight 2 bits  rotational\_position += position\_movement; // Update the absolute position values. (Position of this encoder since CPU reset.)  bitshift\_cur\_pos = current\_position << 2; //Bitshiftleft - The leftmost 2 bits in current\_position are shifted out of existence  bitshift\_last\_pos = last\_position << 2;  bitshift\_pos\_delta = (bitshift\_cur\_pos - bitshift\_last\_pos);  position\_movement = bitshift\_pos\_delta >> 2; //BitshiftRight 2 bits  rotational\_position += position\_movement; // Update the absolute position values. (Position of this encoder since CPU reset.) |
| --- |

11/17/22

* I have two programs now that can publish left\_ticks and right\_ticks
* $ cd catkin\_ws/src
* $ catkin\_create\_pkg tractor\_odom std\_msgs rospy roscpp std\_msgs sensor\_msgs geometry\_msgs tf
* $ cd ~/catkin\_ws
* $ catkin\_make
* $ cd ~/catkin\_ws/src/tractor\_odom
* $ mkdir launch
* $ mkdir scripts
* $ cd ~/catkin\_ws/
* $ catkin\_make --only-pkg-with-deps tractor\_odom
* $ roscd tractor\_odom
* $ gedit CMakeLists.txt
* Remove the hashtag on line 5 to make sure that C++11 support is enabled.
* $ cd launch
* $ sudo nano tractor\_odom.launch

| <launch>  <node pkg="rosserial\_python" type="serial\_node.py" name="serial\_node">  <param name="port" value="/dev/ttyACM0"/>  <param name="baud" value="115200"/>  </node>  <node pkg="rosserial\_python" type="serial\_node.py" name="serial\_node">  <param name="port" value="/dev/ttyACM2"/>  <param name="baud" value="115200"/>  </node>  </launch> |
| --- |

* $ sudo chmod +x tractor\_odom.launch