www.YostLabs.com
1-740-876-4936

AN2020.02 "3-Space Sensor NANO SPI with Arduino"

Introduction

The Yost 3-Space Sensors can communicate over SPI, this allows them to be interfaced by a variety of different devices including Arduino.

The Arduino is a low-cost micro controller and can be used to easily communicate and test out the functionality of the 3-Space Sensor, with minimal setup and difficulty.

This document will include the setup and pin placement for the Arduinos with +5v output pins such as the Arduino Uno or Arduino Mega 2560. The document also contains similar information for the setup of the 3.3v output Arduino Nano, the setup will be slightly different for both the 5v and 3.3v Arduino, but the underlying code examples should remain the same as the underlying SPI calls will be handled by the Arduino built-in SPI.h library.

Description

Before connecting your microcontroller, check the operating voltage of your microcontroller, the TSS-Nano only supports 3.3v. Use of incorrect voltages may result in incorrect readings and/or damage to the sensor. We use the TSS-Nano for this guide and will demonstrate how to connect to both a 3.3v and a 5v Arduino board.

Most Arduino use the same ports for the Arduino SPI library with the exeception of the Ardunio Mega 2560. The ports are listed below with the Mega's ports in parenthesis. For additional information, you can find the Arduino SPI reference guide here: "https://www.arduino.cc/en/reference/SPI".

For the Arduino, make sure the following pins are connected, for ease of reference we will be following a color coded pattern.

MOSI – 11 (51) [Green]

MISO – 12 (50) [Yellow]

SCK – 13 (52) [White]

SS – 10 (53) [Orange]

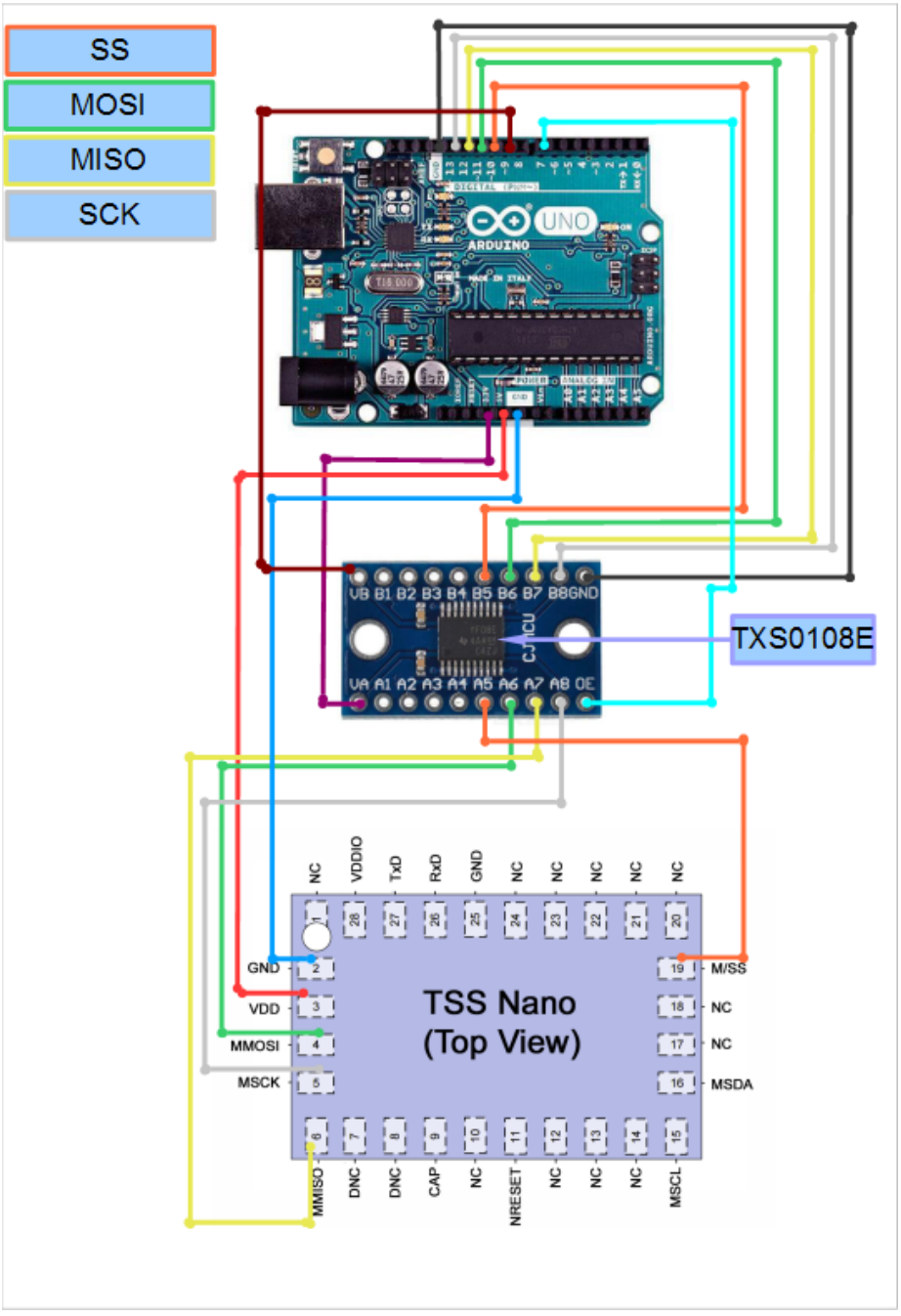
5V – 5V [Red]

3.3V – 3.3V [Purple]

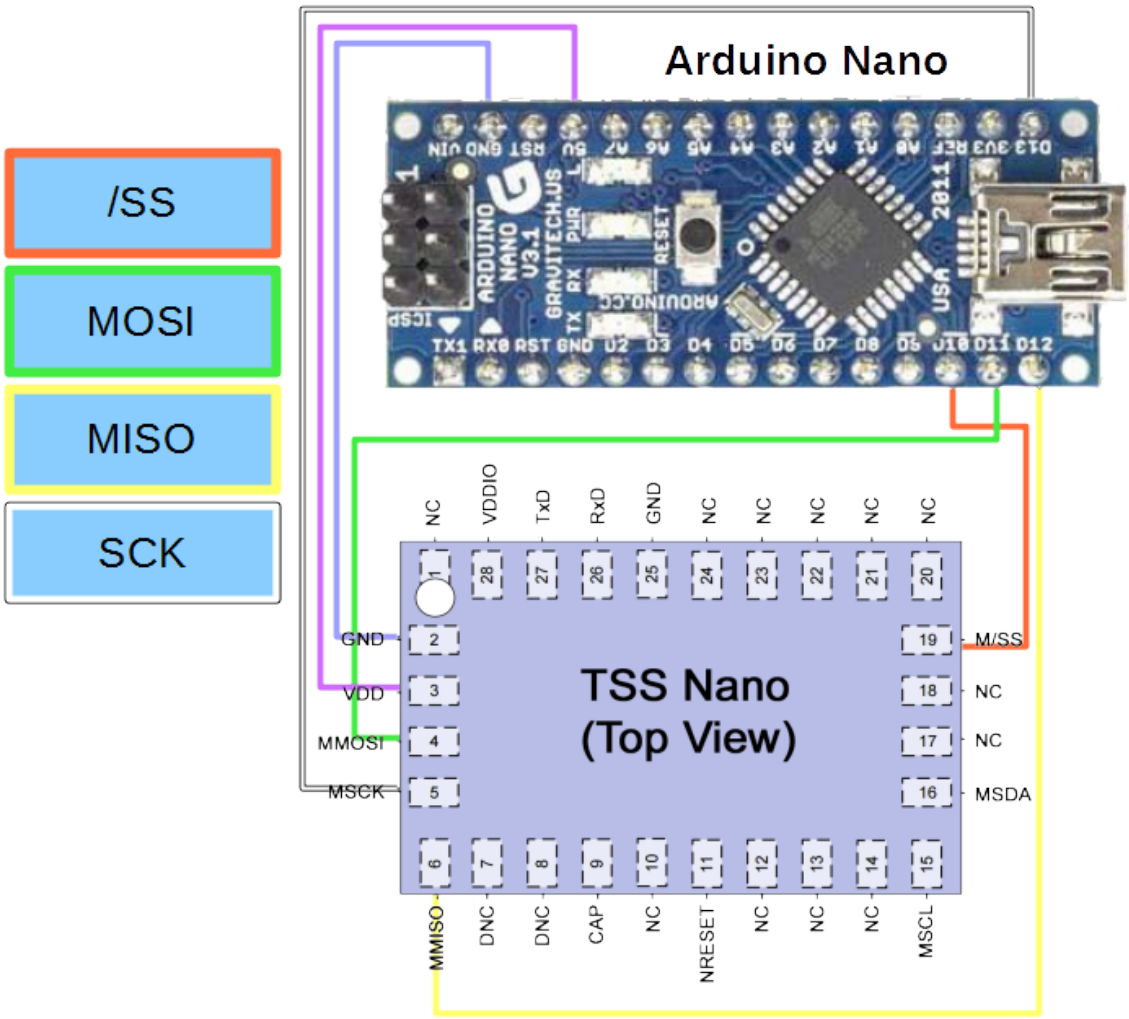
GND – GND [Black, Blue]

Once the pins are connected to the Arduino, the next step will be to step down the voltage to 3.3v so that it can be safely used in the 3-Space-Sensor, to do this we will be using the TXS0108E, which is a bi-directional level shifter that will turn the 5v arduino output pins into compatible 3.3v for the sensor.

The bi-directional shifter will need 5v and 3.3v power on both ends, and if the sensor will be powered via the vin pin, then another 3-5v output pin will need to be dedicated to it, otherwise a usb cable will suffice to power the device. The data and setup pdf for the TXS0108E can be found here, "http://www.icbase.com/File/PDF/TTI/TTI43821108.pdf".



If you are using a 3.3v Arudino, there is no need for a level shifter and the two boards can be connected directly.



This program, when ran, will print out the sensor's serial number. This number can be verified with either the sticker on the sensor or the serial number you can find in the 3-Space Suite.

Complete Example:

```
#include<SPI.h>
#include<stdlib.h>
#define TRANSACTION_SETTINGS SPISettings(1000000,MSBFIRST,SPI_MODE0)
#define SS 10
// Extra pins for use with the level shifter
#define OE 7
#define Vout 9
//these are the most reliable delays tested
#define MicroDelay 6
#define MiliDelay 3
/*Spi Command List:
      0xE9: This command prepares the sensor to receive a TSS command
      0xD9: This command will clear the command stored in the sensor
      0x69: This command tells the sensor to return the command data
      0x81: This command prepares the sensor to send the status of the slave device
      more information in data sheet page 23 */
void setup() {
  Serial.begin(19200);
  SPI.begin();
  pinMode(OE, OUTPUT);
  digitalWrite(OE, HIGH);
  pinMode(Vout, OUTPUT);
  digitalWrite(Vout, HIGH);
  pinMode(SS,OUTPUT);
  digitalWrite(SS,HIGH);
  SPI.beginTransaction(TRANSACTION_SETTINGS);
  ReadSerialNumber();
}

void loop() {
  //print out tared orientation every half second
  delay(500);
  ReadOrientationQuat();
}

void ReadSerialNumber(){
  char buff[32];

  digitalWrite(SS,LOW);
  SPI.transfer(0xe9); //tell the sensor the next byte will be a command
  delayMicroseconds(MicroDelay);
  SPI.transfer(0xed); //command to get Serial Number of sensor
  digitalWrite(SS,HIGH);
  delay(MiliDelay);
  unsigned char in_byte[5];
  unsigned char polling_byte = 0x00;
  //Poll the sensor until it returns the return data length in this case it will be
4
  while(polling_byte != 4){
    digitalWrite(SS,LOW);
    polling_byte = SPI.transfer(0x69);
    delayMicroseconds(MicroDelay);
```

```
    digitalWrite(SS,HIGH);
    delay(MiliDelay);
  }
  //Read out requested command data
  digitalWrite(SS,LOW);
  for(unsigned int i = 0; i < polling_byte; i++){
    in_byte[i]=SPI.transfer(0xff);
    delayMicroseconds(MicroDelay);
  }
  digitalWrite(SS,HIGH);
  sprintf(buff,"%02x%02x%02x%02x",in_byte[0],in_byte[1],in_byte[2],in_byte[3]);
  Serial.println(buff);
  Serial.println("done");

}
void ReadOrientationQuat(){
  char buff[32];
  byte in_byte[16];
  unsigned char polling_byte = 0x00;
  digitalWrite(SS,LOW);
  SPI.transfer(0xe9); //tell the sensor the next byte will be a command
  delayMicroseconds(MicroDelay);
  SPI.transfer(0x00); //command to get tared orientation in quaternion form
  digitalWrite(SS,HIGH);
  delay(MiliDelay);
  //Poll the sensor until it returns the return data length
  while(polling_byte != 16){
    digitalWrite(SS,LOW);
    polling_byte = SPI.transfer(0x69);
    delayMicroseconds(MicroDelay);
    digitalWrite(SS,HIGH);
    delay(MiliDelay);
  }
  for(unsigned int i = 0; i < polling_byte; ++i){
    //Read out requested command data
    digitalWrite(SS,LOW);
    in_byte[i]=SPI.transfer(0xff);
    delayMicroseconds(MicroDelay);
  }
  for(unsigned int i = 0; i < 4; ++i){
    //reverse endianess and append to string
    ((float*)in_byte)[i] = ReverseFloat(((float*)in_byte)[i]);
    dtostrf(((float*)in_byte)[i],8,5,(buff +(i*8 +i)));
    if(i*8 + i-1 < 32 && i){
      buff[i*8 + i-1] = ',';
    }
  }
  Serial.println(buff);
}

float ReverseFloat( const float inFloat )
{
   float retVal;
   char *floatToConvert = ( char* ) & inFloat;
   char *returnFloat = ( char* ) & retVal;

   // swap the bytes into a temporary buffer
```

```
    returnFloat[0] = floatToConvert[3];
    returnFloat[1] = floatToConvert[2];
    returnFloat[2] = floatToConvert[1];
    returnFloat[3] = floatToConvert[0];
    return retVal;
}
```