```
//
//  Working Version V2
//



/* Written by Michael Welsh September 7, 2017 for implementation using the Arduino UNO
 * To be used with complementary program receiver_sample_JMW_V1
 *
 * This program is designed to integrate the UNO and nRF905.  It operates in a
 * TRANSMIT mode to send data to the receiver.  The receiver is also integrated to
 * a UNO and NRF905.  This program sends up to 16 integer data types (2 bytes each) for a
 * maximum 32 byte packet.  Range for integer is -32768 through 32767.
 *
 *
 * It is the goal to develop a program which integrates a UNO with a
 * MPU6050, and NRF905 transmitter in order to send acceleration data to the receiver
 * and then to a Processing program which will graphically present the data
 * as a vehicle in a 3 dimensional space.
 *
 * Program and Library adopted from the Rethink Tech Inc. - Tinkbox (nRF905) and the
 * Jeff Rowberg I2C device class using the Digital Motion Processing (DMP), MotionApps 2.0,
 * (MPU6050_DMP6).
 *
 * This program uses 3,908 Bytes (12%) on the UNO.
 *
 * UNO to nRF905 BOARD PIN and Control Feature
 *
 * 7 -> CE     Standby - High = TX/RX mode, Low = Standby
 * 8 -> PWR    Power Up - High = On, Low = Off
 * 9 -> TXE    TX or RX mode - High = TX, Low = RX
 * 2 -> CD     Carrier Detect - High when RF signal detected, for collision avoidance
 * 3 -> DR     Data Ready - High when finished transmitting/data received
 * 10 -> CSN   SPI SS
 * 12 -> SO    SPI MISO
 * 11 -> SI    SPI MOSI
 * 13 -> SCK   SPI SCK
 * GND -> GND  Ground

  Note on use of the CD Pin.  I have completed a test in which the trasnmitter
  was able to send data to the Receiver with this pin disconnected.  This may be usefull
  because this pin may be required when using the MPU6050 which uses Pin 2 of the UNO
  for interrupt processing.  Both the transmitter and receiver CD connection is
  optional for this program.
*/


int data = A0;
```

```
#include <nRF905.h>  //Library Author Zak Kemble, Web: http://blog.zakkemble.co.uk/nrf905-
avrarduino-librarydriver/
#include <SPI.h>  //SPI Master Library for Arduino

/*
   Note in selection of the RXADDR and TXADDR.  Nordic recommends that the address length be 24 bits
or higher
   in length.   Smaller lengths can lead to statistical failures due to the address bein repeated as part of
the
   data packet.  Each byte should be unique.  The address should have several level shifts (101010101).
*/

#define RXADDR {0xFE, 0x4C, 0xA6, 0xE5} // Address of this device (4 bytes)
#define TXADDR {0x58, 0x6F, 0x2E, 0x10} // Address of device to send to (4 bytes)

#define PACKETPAUSE 250 // Short Break after sending each data packet

void setup()
{
	// Power up nRF905 and initialize
	nRF905_init();

	// Send this device address to nRF905
	byte incoming[] = RXADDR;
	nRF905_setRXAddress(incoming);

  // Send remote device address to nRF905
  byte outgoing[] = TXADDR;
  nRF905_setTXAddress(outgoing);

	// Put into receive mode
	nRF905_receive();

  // Start serial coomunication with monitor.  Send start message.
	Serial.begin(9600);
	Serial.println(F("Transmitter started"));
}

void loop()
{
/*  Put data to transmit in the data array.  Maximum packet data size is 32 bytes. The Arduino UNO
    has 2 byte integers. Since an integer data type has 2 bytes (16 bits), only 16 integer data items
    can be sent in a packet.  Any data beyond 32 bytes will be dropped and not readable at the receiver.
    Additional data must be sent in the next Packet.

    NRF905_MAX_PAYLOAD is defined in the nRF905 library as 32 bytes.
*/
```

```
  int data[NRF905_MAX_PAYLOAD] = {15, 25, 30, 45, 48, -32767, 12, 70, -256, 96, 100, 110, 185, 255,
256, 32767};
  nRF905_setData(data, sizeof(data));

/*  Debug printout of data array showing information transmitted.

  for (int i=0; i<=15; i++)
        {
          Serial.print("   ");
          Serial.print(data[i], DEC);
          delay(100);
        }
  Serial.println();
  Serial.println();
*/

              // Send data array payload (send fails if other transmissions are going on, keep trying until
success)
  while(!nRF905_send());

              // Put into receive mode - I think the nRF905 goes into the Receive mode after Transmitting the
packet
  // if the receive mode was initialized as indicated in the setup().

  delay(PACKETPAUSE);
```