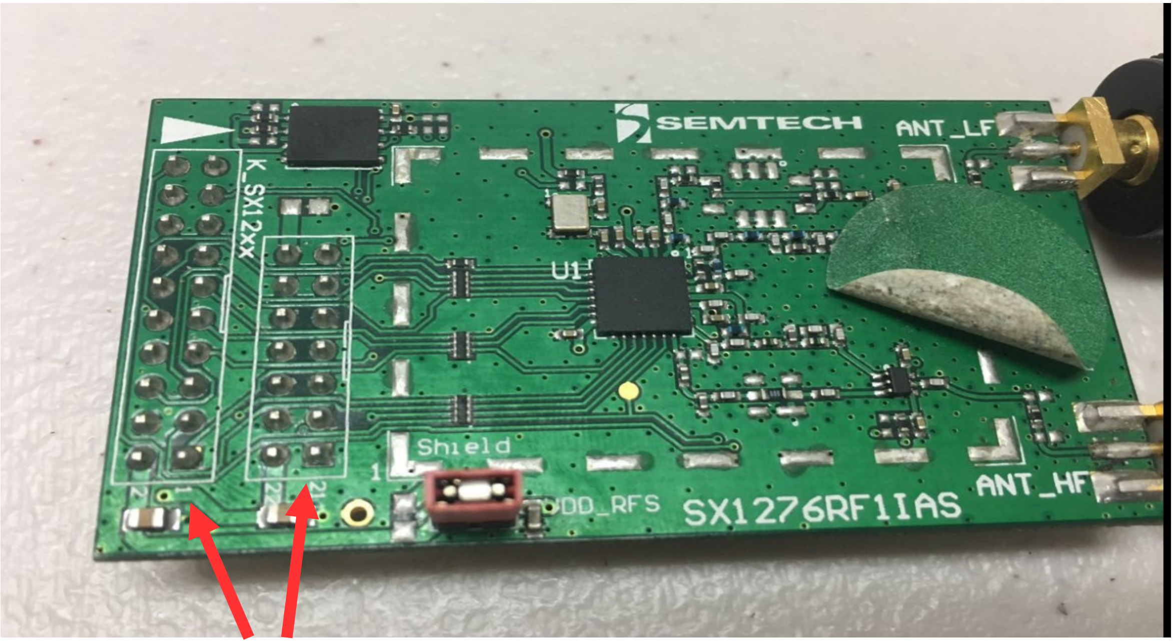
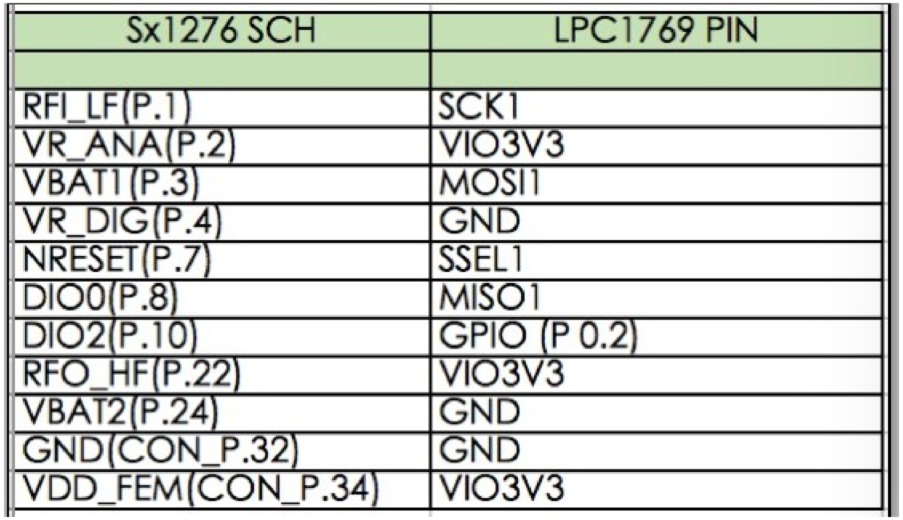
Lora RF Module





Sx1276 SCH	LPC1769 PIN
RFI_LF(P.1)	SCK1
VR_ANA(P.2)	VIO3V3
VBAT1(P.3)	MOSI1
VR_DIG(P.4)	GND
NRESET(P.7)	SSEL1
DIO0(P.8)	MISO1
DIO2(P.10)	GPIO (P 0.2)
RFO_HF(P.22)	VIO3V3
VBAT2(P.24)	GND
GND(CON_P.32)	GND
VDD_FEM(CON_P.34)	VIO3V3

Harry Li, Ph.D.

Sample Code for Lora RF Module

```
struct wheelsToScale wheelScale;

int main(void) {

    unifiedMotorInit();
    LoRabegin(915000000);

    for (;;) {

        rfcontrollerprocessing();
    }
}
void rfcontrollerprocessing() {
    packetSize = parsePacket(0);
    if (packetSize) {
        while (available()) {
            emergencyStop = false;
            receiveData = read();
            //printf("%c",receiveData);
            //Hat Switch control
            switch (receiveData) {
            case 'O':
                AGVStop();
                isGoingBackwards = false;
                isGoingForwards = false;
                break;
            case 'X':
            case 'Y':
            case 'Z':
```

```
        if (!isGoingForwards) {
            isGoingBackwards = true;
            if (receiveData == 'X') {
                //                 RIGHT TURN
                AGVGoBackward(9, 12);
            } else if (receiveData == 'Y') {
                AGVGoBackward(10, 10);
            } else if (receiveData == 'Z') {
                //                 LEFT TURN
                AGVGoBackward(12, 9);
            }
        }
        break;
```

Acknowledgement: Thanks for CTI One Corporation make this code available for Dr. Harry Li for CMPE 245 use.

Sample Code for Lora RF Module

```
case 'C':
                AGVStop();
                isGoingBackwards = false;
                isGoingForwards = false;
                emergencyStop = true;
                trueStart = false;
                break;
            case 'A':
            case 'B':
            case 'D':
            case 'E':
            case 'F':
                if (!isGoingBackwards) {
                    isGoingForwards = true;
                    if (receiveData == 'A') {
                        AGVStraightForward(current_speed, current_speed);
                        trueStart = true;
                    } else if (receiveData == 'B') {
//                  speed up by 20% and speed down by 10%
                        AGVRightTurn(current_speed + (current_speed * 1 /
5),
                                current_speed - (current_speed * 1 / 10));
                        trueStart = true;
                    } else if (receiveData == 'D') {
//                  speed up by 20% and speed down by 10%
                        AGVLeftTurn(current_speed - (current_speed * 1 / 10),
                                current_speed + (current_speed * 1 / 5));
                        trueStart = true;
```

Harry Li, Ph.D.

Sample Code for Lora RF Module

```
        } else if (receiveData == 'E') {
                        if (current_speed <= 10 * SPEED_SHIFT) //scaling -- Maximum speed will be 30
                            current_speed += SPEED_SHIFT;   //scaling and speeding up
                        if (trueStart)
                            AGVStraightForward(current_speed, current_speed);
                    } else if (receiveData == 'F') {
                        //
                        if (current_speed >= SPEED_SHIFT)
                            current_speed -= SPEED_SHIFT;
                        if (current_speed < 10)
                            current_speed = 10;
                        if (trueStart)
                            AGVStraightForward(current_speed, current_speed);
                    }
                }
                break;
                default:
                //other button
                break;
            }
        }

    }
}
```

Sample Code for Lora RF Module 1

```
/*
 * LoRa.c
 *
 *  Created on: Oct 29, 2016
 * LoRa.cpp
 *
 *  Created on: Oct 29, 2016
 */
#include <stdio.h>
#include <stddef.h>
#include "LoRa.h"
#include "ssp.h"
#include "extint.h"
// registers
#define REG_FIFO                 0x00
#define REG_OP_MODE              0x01
#define REG_FRF_MSB              0x06
#define REG_FRF_MID              0x07
#define REG_FRF_LSB              0x08
#define REG_PA_CONFIG            0x09
#define REG_LNA                  0x0c
#define REG_FIFO_ADDR_PTR        0x0d
#define REG_FIFO_TX_BASE_ADDR    0x0e
#define REG_FIFO_RX_BASE_ADDR    0x0f
#define REG_FIFO_RX_CURRENT_ADDR 0x10
#define REG_IRQ_FLAGS            0x12
#define REG_RX_NB_BYTES          0x13
#define REG_PKT_RSSI_VALUE       0x1a
#define REG_PKT_SNR_VALUE        0x1b
```

```
#define REG_MODEM_CONFIG_1       0x1d
#define REG_MODEM_CONFIG_2       0x1e
#define REG_PREAMBLE_MSB         0x20
#define REG_PREAMBLE_LSB         0x21
#define REG_PAYLOAD_LENGTH       0x22
#define REG_RSSI_WIDEBAND        0x2c
#define REG_DETECTION_OPTIMIZE   0x31
#define REG_DETECTION_THRESHOLD  0x37
#define REG_SYNC_WORD            0x39
#define REG_DIO_MAPPING_1        0x40
#define REG_VERSION              0x42
// modes
#define MODE_LONG_RANGE_MODE     0x80
#define MODE_SLEEP               0x00
#define MODE_STDBY               0x01
#define MODE_TX                  0x03
#define MODE_RX_CONTINUOUS       0x05
#define MODE_RX_SINGLE           0x06

// PA config
#define PA_BOOST                 0x80

// IRQ masks
#define IRQ_TX_DONE_MASK           0x08
#define IRQ_PAYLOAD_CRC_ERROR_MASK 0x20
#define IRQ_RX_DONE_MASK           0x40
```

Sample Code for Lora RF Module 2

```
#define MAX_PKT_LENGTH          255
#define      LOW 0
#define HIGH      1

#define function_not_required 0

int _packetIndex=0;
int _implicitHeaderMode=0;
int _frequency=0;
#if function_not_required
void (*_onReceive)(int);
#endif
char receiveLora=0;

#if function_not_required
void onReceivedata(int packetSize) {
  // received a packet
  printf("Received packet ");
  int i=0;
  // read packet
  for (i = 0; i < packetSize; i++) {
    printf("%c\n",(char)read());
  }

  // print RSSI of packet
  print("' with RSSI ",packetRssi());
}
```

```
void EINT3_IRQHandler(void)
{
        LPC_SC->EXTINT = EINT3;          /* clear interrupt */
        printf("Interrupt triggered\n");
        // Toggle Led On
        //LPC_GPIO0->FIOPIN ^= (1<<2);

        handleDio0Rise();
        /*Clear interrupts */
        LPC_GPIOINT->IO0IntClr = 0xFFFFFFFF;
}
#endif

void gpioInit()
{
        // Select P0.2 as GPIO for RESET
        LPC_PINCON->PINSEL0 &= ~(3<<4);

        // P0.3 as GPIO
        LPC_PINCON->PINSEL0 &= ~(3<<6);
        // P0.2 as output For RESET
        LPC_GPIO0->FIODIR |= (1<<2);
        // P0.3 as input For DIO0
        //LPC_GPIO0->FIODIR &= ~(1<<3);
        //LPC_GPIOINT->IO0IntEnR |= (1<<3);
        //NVIC_EnableIRQ(EINT3_IRQn);
}
```

Sample Code for Lora RF Module 3

```
void digitalWrite(uint8_t pin, uint8_t value)
{
//      printf("Pin : %d, value %d\n",pin,value);
        if(value == 1)
        {
                LPC_GPIO0->FIOPIN |= (1<<pin);
        }
        else if(value == 0)
        {
                LPC_GPIO0->FIOPIN &= ~(1<<pin);
        }
}
int LoRabegin(long frequency)
{
  // setup pins
        uint8_t version =0;
        int i=0;
        gpioInit();
  // perform reset
  digitalWrite(LORA_DEFAULT_RESET_PIN, LOW);
  for(i=0;i<100000;i++);
  digitalWrite(LORA_DEFAULT_RESET_PIN, HIGH);
  for(i=0;i<10000000;i++);
```

```
  // set SS high
  CHIP_DESELECT();
  //digitalWrite(_ss, HIGH);
  SSP1Init();
  // start SPI
  //SPI.begin();
  // check version
  version = readRegister(REG_VERSION);
  //  printf("Version is %x\n",version);
  if (version != 0x12) {
    return 0;
  }
  // put in sleep mode
  sleep();
  // set frequency
  setFrequency(frequency);
  // set base addresses
  writeRegister(REG_FIFO_TX_BASE_ADDR, 0);
  writeRegister(REG_FIFO_RX_BASE_ADDR, 0);
  // set LNA boost
  writeRegister(REG_LNA, readRegister(REG_LNA) | 0x03);
  // set output power to 17 dBm
  setTxPower(17);
  // put in standby mode
  idle();
  return 1;
}
```

Harry Li, Ph.D.

Sample Code for Lora RF Module 4

```
void end()
{
  // put in sleep mode
  sleep();

  // stop SPI
  //SPI.end();
}

int LoRabeginPacket(int implicitHeader)
{
  // put in standby mode
  idle();

  if (implicitHeader)
  {
    implicitHeaderMode();
  }
  else {

    explicitHeaderMode();
  }

  // reset FIFO address and paload length
  writeRegister(REG_FIFO_ADDR_PTR, 0);
  writeRegister(REG_PAYLOAD_LENGTH, 0);

  return 1;
}
```

```
int LoRaendPacket()
{
  uint8_t rOut=0;
  // put in TX mode
  writeRegister(REG_OP_MODE, MODE_LONG_RANGE_MODE | MODE_TX);
  // wait for TX done
   while((readRegister(REG_IRQ_FLAGS) & IRQ_TX_DONE_MASK) == 0);
  // clear IRQ's
  writeRegister(REG_IRQ_FLAGS, IRQ_TX_DONE_MASK);
  return 1;
}

int parsePacket(int size)
{
        int packetLength = 0;
        int irqFlags = readRegister(REG_IRQ_FLAGS);
        if (size > 0)
        {
                implicitHeaderMode();
                writeRegister(REG_PAYLOAD_LENGTH, size & 0xff);
        }
        else
        {
                explicitHeaderMode();
        }

        // clear IRQ's
        writeRegister(REG_IRQ_FLAGS, irqFlags);
```

Sample Code for Lora RF Module 5

```
    if ((irqFlags & IRQ_RX_DONE_MASK) && (irqFlags & IRQ_PAYLOAD_CRC_ERROR_MASK) == 0)
    {
        // received a packet
        _packetIndex = 0;
        // read packet length
        if (_implicitHeaderMode)
        {
            packetLength = readRegister(REG_PAYLOAD_LENGTH);
        }
        else
        {
            packetLength = readRegister(REG_RX_NB_BYTES);
        }
        // set FIFO address to current RX address
        writeRegister(REG_FIFO_ADDR_PTR, readRegister(REG_FIFO_RX_CURRENT_ADDR));
        // put in standby mode
        idle();
    }
    else if (readRegister(REG_OP_MODE) != (MODE_LONG_RANGE_MODE | MODE_RX_SINGLE))
    {
        // not currently in RX mode
        // reset FIFO address
        writeRegister(REG_FIFO_ADDR_PTR, 0);
        // put in single RX mode
        writeRegister(REG_OP_MODE, MODE_LONG_RANGE_MODE | MODE_RX_SINGLE);
    }
    return packetLength;
}
```

Sample Code for Lora RF Module 6

```
int packetRssi()
{
    return (readRegister(REG_PKT_RSSI_VALUE) - (_frequency < 868E6 ? 164 : 157));
}
float packetSnr()
{
    return ((int8_t)readRegister(REG_PKT_SNR_VALUE)) * 0.25;
}
size_t writebyte(uint8_t byte)
{
    return write(&byte, sizeof(byte));
}
size_t write(const uint8_t *buffer, size_t size)
{
    int currentLength = readRegister(REG_PAYLOAD_LENGTH);
    size_t i=0;
    // check size
    if ((currentLength + size) > MAX_PKT_LENGTH)
    {
        size = MAX_PKT_LENGTH - currentLength;
    }
    // write data
    for (i = 0; i < size; i++)
    {
        writeRegister(REG_FIFO, buffer[i]);
    }
    // update length
    writeRegister(REG_PAYLOAD_LENGTH, currentLength + size);
    return size;
}
```

Sample Code for Lora RF Module 7

```
int available()
{
        return (readRegister(REG_RX_NB_BYTES) - _packetIndex);
}
int read()
{
        if (!available()) {
                return -1;
        }
        _packetIndex++;
        return readRegister(REG_FIFO);
}

int peek()
{
        if (!available()) {
                return -1;
        }
        // store current FIFO address
        int currentAddress = readRegister(REG_FIFO_ADDR_PTR);
        // read
        uint8_t b = readRegister(REG_FIFO);
        // restore FIFO address
        writeRegister(REG_FIFO_ADDR_PTR, currentAddress);
        return b;
}
```

Sample Code for Lora RF Module 8

```
void flush()
{
}

#if function_not_required
void onReceive(void(*callback)(int))
{
        _onReceive = callback;
        //writeRegister(REG_DIO_MAPPING_1, 0x00);
        if (callback)
        {
                writeRegister(REG_DIO_MAPPING_1, 0x00);

                //attachInterrupt(digitalPinToInterrupt(_dio0), onDio0Rise, RISING);
        }
        else
        {
                //detachInterrupt(digitalPinToInterrupt(_dio0));
        }
}
```

Sample Code for Lora RF Module 9

```
void idle()
{
	writeRegister(REG_OP_MODE, MODE_LONG_RANGE_MODE | MODE_STDBY);
}

void sleep()
{
	writeRegister(REG_OP_MODE, MODE_LONG_RANGE_MODE | MODE_SLEEP);
}

void setTxPower(int level)
{
	if (level < 2)
	{
		level = 2;
	}
	else if (level > 17)
	{
		level = 17;
	}

	writeRegister(REG_PA_CONFIG, PA_BOOST | (level - 2));
}
```

Sample Code for Lora RF Module 10

```
void setFrequency(long frequency)
{
        _frequency = frequency;
//      printf("frequency is %d,%d\n",frequency,_frequency);
        uint64_t frf = ((uint64_t)frequency << 19) / 32000000;
        writeRegister(REG_FRF_MSB, (uint8_t)(frf >> 16));
        writeRegister(REG_FRF_MID, (uint8_t)(frf >> 8));
        writeRegister(REG_FRF_LSB, (uint8_t)(frf >> 0));
}
void setSpreadingFactor(int sf)
{
        if (sf < 6)
        {
                sf = 6;
        }
        else if (sf > 12)
        {
                sf = 12;
        }
        if (sf == 6)
        {
                writeRegister(REG_DETECTION_OPTIMIZE, 0xc5);
                writeRegister(REG_DETECTION_THRESHOLD, 0x0c);
        }
        else
        {
                writeRegister(REG_DETECTION_OPTIMIZE, 0xc3);
                writeRegister(REG_DETECTION_THRESHOLD, 0x0a);
        }
        writeRegister(REG_MODEM_CONFIG_2, (readRegister(REG_MODEM_CONFIG_2) & 0x0f) | ((sf << 4) & 0xf0));
}
```

Sample Code for Lora RF Module 11

```
void setSignalBandwidth(long sbw)
{
  int bw;
  if (sbw <= 7.8E3) {
    bw = 0;
  } else if (sbw <= 10.4E3) {
    bw = 1;
  } else if (sbw <= 15.6E3) {
    bw = 2;
  } else if (sbw <= 20.8E3) {
    bw = 3;
  } else if (sbw <= 31.25E3) {
    bw = 4;
  } else if (sbw <= 41.7E3) {
    bw = 5;
  } else if (sbw <= 62.5E3) {
    bw = 6;
  } else if (sbw <= 125E3) {
    bw = 7;
  } else if (sbw <= 250E3) {
    bw = 8;
  } else /*if (sbw <= 250E3)*/ {
    bw = 9;
  }
  writeRegister(REG_MODEM_CONFIG_1, (readRegister(REG_MODEM_CONFIG_1) & 0x0f) | (bw << 4));
}
```

Sample Code for Lora RF Module 12

```
void setCodingRate4(int denominator)
{
  if (denominator < 5) {
    denominator = 5;
  } else if (denominator > 8) {
    denominator = 8;
  }

  int cr = denominator - 4;

  writeRegister(REG_MODEM_CONFIG_1, (readRegister(REG_MODEM_CONFIG_1) & 0xf1) | (cr << 1));
}

void setPreambleLength(long length)
{
  writeRegister(REG_PREAMBLE_MSB, (uint8_t)(length >> 8));
  writeRegister(REG_PREAMBLE_LSB, (uint8_t)(length >> 0));
}

void setSyncWord(int sw)
{
  writeRegister(REG_SYNC_WORD, sw);
}
```

Sample Code for Lora RF Module 13

```
void crc()
{
  writeRegister(REG_MODEM_CONFIG_2, readRegister(REG_MODEM_CONFIG_2) | 0x04);
}

void noCrc()
{
  writeRegister(REG_MODEM_CONFIG_2, readRegister(REG_MODEM_CONFIG_2) & 0xfb);
}

uint8_t random()
{
  return readRegister(REG_RSSI_WIDEBAND);
}
void explicitHeaderMode()
{
  _implicitHeaderMode = 0;
  writeRegister(REG_MODEM_CONFIG_1,
readRegister(REG_MODEM_CONFIG_1) & 0xfe);
}
```

Sample Code for Lora RF Module 14

```
void implicitHeaderMode()
{
  _implicitHeaderMode = 1;
  writeRegister(REG_MODEM_CONFIG_1, readRegister(REG_MODEM_CONFIG_1) | 0x01);
}
#if function_not_required
void handleDio0Rise()
{
  int irqFlags = readRegister(REG_IRQ_FLAGS);
  int i=0;
  // clear IRQ's
  writeRegister(REG_IRQ_FLAGS, irqFlags);
  if ((irqFlags & IRQ_PAYLOAD_CRC_ERROR_MASK) == 0) {
    // received a packet
    _packetIndex = 0;
    // read packet length
    int packetLength = _implicitHeaderMode ? readRegister(REG_PAYLOAD_LENGTH) :
readRegister(REG_RX_NB_BYTES);
    // set FIFO address to current RX address
    writeRegister(REG_FIFO_ADDR_PTR, readRegister(REG_FIFO_RX_CURRENT_ADDR));
//    if (_onReceive) {
//      _onReceive(packetLength);
//    }
    for (i = 0; i < packetLength; i++)
    {
         receiveLora = read();
         printf("Receive data is %c\n",receiveLora);
     }
    // reset FIFO address
    writeRegister(REG_FIFO_ADDR_PTR, 0);
  }
}
#endif
```

Sample Code for Lora RF Module 15

```
uint8_t readRegister(uint8_t address)
{
  return singleTransfer(address & 0x7f, 0x00);
}

void writeRegister(uint8_t address, uint8_t value)
{

      singleTransfer(address | 0x80, value);
}
uint8_t singleTransfer(uint8_t address, uint8_t value)
{
  uint8_t response=0;

  //digitalWrite(_ss, LOW);
  CHIP_SELECT();
  //SPI.beginTransaction(_spiSettings);
  response = ssp1Transfer(address);
  //printf("response %x\n",response);
  response = ssp1Transfer(value);
  //printf("response %x\n",response);
  CHIP_DESELECT();
  //digitalWrite(_ss, HIGH);
  return response;
}
```

```
#if function_not_required
void onDio0Rise()
{
      handleDio0Rise();
}
#endif
```