

Developer Kit Project

Remote temperature sensor

Project overview

The aim of this project is to build a hydrogen powered remote temperature sensor.

It is based on the Arduino, Developer Kit fuel cell shield, Maxim DS18B20 1 Wire temperature sensor, and the Hope RFM12 radio transceiver.



Table Of Contents

HARDWARE REQUIREMENTS	2
SOFTWARE REQUIREMENTS	2
SOFTWARE PREPARATION	2
JeeLib library	
Developer Kit library	
HARDWARE PREPARATION	2
RFM12	
Testing the radios	
BUILDING THE SENDER	4
PROGRAMMING THE RECEIVER	7
EXPANSION IDEAS	7

Hardware requirements

- 2 Arduinos
- Developer Kit 3W fuel cell inventor kit
- 2 Hope 433Mhz RFM12 transceivers
- 2 RFM12 breakout boards (optional)
- 1 DS18B20 temperature sensor
- Bread board
- Wires
- 5k resistor

Software requirements

- Arduino IDE
- The Developer Kit library
- The JeeLib library for RFM12 transceivers

Software preparation

JeeLib library

The JeeLib library is a set of utilities for running JeeNodes, which are small arduino compatible boards with a built in RFM12 radio. The library gives us an easy way to run these radios. We need the new version because it adds support for pin change interrupts. Download the library [here](#). Then unzip it in your sketchbook/libraries folder.

Developer Kit library

Make sure you've got the latest version of the Developer Kit Arduino library installed by downloading and unzipping into your sketchbook/libraries folder.

Start the Arduino IDE and have a look at the file->examples->Developer Kit->sender and receiver sketches.

Hardware preparation

RFM12

There are quite a few RFM12 radio module options. There are 315MHZ, 433MHZ, 868MHz and 915MHz versions. We're using the 433MHz version. The others may work but haven't been tested in this application. Also, there are mounting differences: some with pins, some without. Either way, we need to be able to plug these into a breadboard. So if you feel happy soldering the 2mm pitch pins then solder some wires onto the following pins: GND, INT, SDO, SCK, SEL, SDI and VCC. Alternatively, you can buy an RFM12 on a breakout board from a number of suppliers, including JeeLib (includes the RFM12) or Robotmotic's breakout board (requires the pinless version of the RFM12). Finally, if you want to make your own breakout board, here are the Eagle schematics for a breakout board (requires the pinned version of the RFM12).

Testing the radios

Start by making the following connections from the radio to an Arduino:

- GND to GND
- VCC to 5V
- INT to pin 2
- SEL to pin 10
- SCK to pin 13
- SDI to pin 11
- SDO to pin 12

Then load the examples->JeeLib->RF12->RF12demo sketch. Download this to the Arduino and open the serial monitor. You should see a lot of information presented regarding the options available and the current state of the RFM12. If you don't see this information then double check your connections, serial port speed (57600). If you edited the JeeLib library with the Arduino IDE open you'll need to restart it.

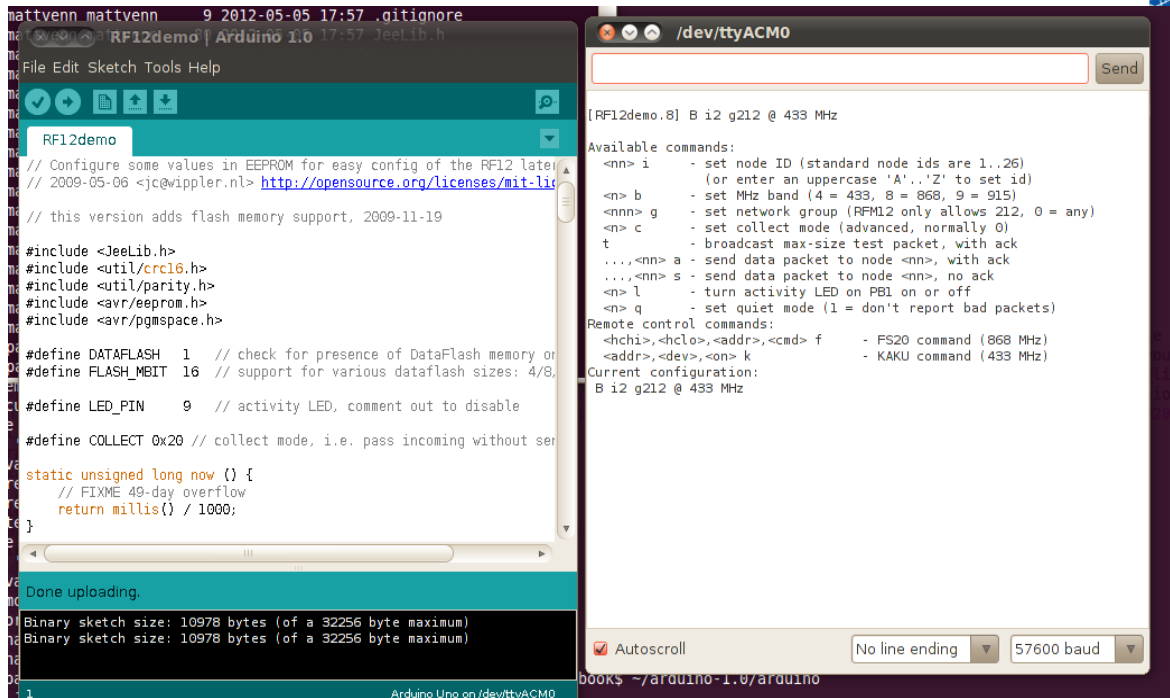


Figure 1. RF12demo

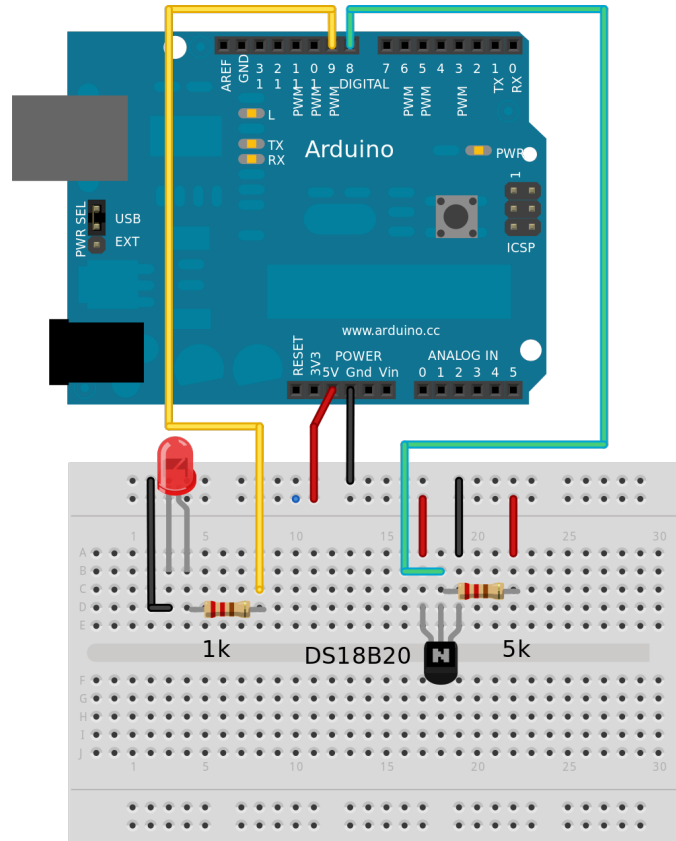
Repeat the steps above to connect the second RFM12 to the second Arduino. Once you have got both Arduinos setup with radios and you've checked that the RM12demo sketch works with both, load the examples->JeeLib->RF12->pingpong sketch. Change the RF12_868MHZ to RF12_433MHZ and then upload the sketch onto both Arduinos. Look at the serial monitor, and if all is working you should see this OK 72101108108111330 repeated every 3 seconds. This shows the Arduinos are talking to each other over the radio. When this is working, we're most of the way there, so celebrate with your favourite beverage!

Building the sender

Let's start with the sender. Remove the wires for the RFM12 and plug the Developer Kit shield onto the Arduino.

Then add the RM12 wires back where they were. It's worth checking that the radio is still working so plug the Arduino in and have a look at the serial monitor. Check that the pingpong sketch is still running as before.

Now let's add an LED to tell us when the data is sent, and the temperature sensor. Use the picture below to wire up the components.



Made with  Fritzing.org

Figure 2. LED and Temperature sensor

Then load the examples->Developer Kit->sender sketch and upload this onto the Arduino. Check the serial monitor. If you have not connected hydrogen to the shield, then you will get lots of errors about the stack voltage being too low. Ignore those and look for information about the temperature and the data being sent every 5 seconds:

```
data ready to send:
temp C: 19.19
stack V: 0.00
stack I: -13.51
sent
```

You should also notice the LED flashing every 5 seconds as the data is sent. If you are not getting a reading for the temperature, check that you've got the temperature sensor wired up correctly, and that there aren't any error messages about the temperature sensor in the serial monitor.

Finally, connect your hydrogen. Check the serial monitor to see that the Developer Kit shield is doing it's job:

```
1.43V, 0.50A
1.46V, 0.50A
1.43V, 0.57A
1.46V, 0.50A
data ready to send:
temp C: 19.25
stack V: 1.44
stack I: 0.51
sent
```

Then unplug the USB cable. The status light of the Developer Kit shield should be flashing on and off every second, and the sent LED should flash every 5 seconds as data is sent. If all the lights go out when you unplug the USB cable, check the hydrogen is connected properly.

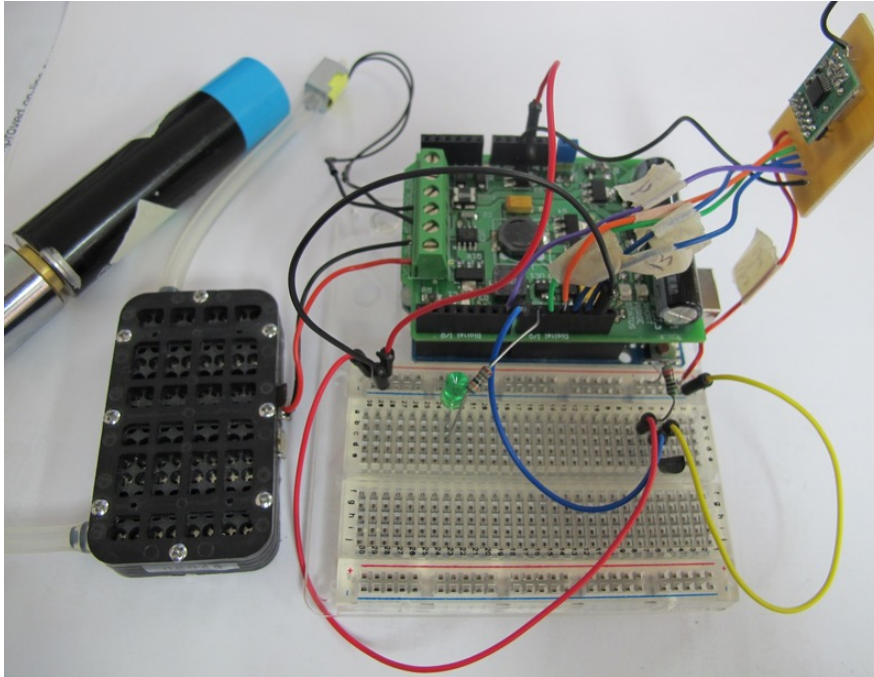


Figure 3. Sender

Note the interrupt line in this case is connected to pin 7. However the hardware has changed and now the interrupt connects to pin 2.

Programming the receiver

Load the examples->Developer Kit->receiver sketch and upload this onto the other Arduino. Check the serial monitor. You should see the data being received every 5 seconds when the sender's LED flashes.

```
receiver start  
rf12 setup done  
got data:  
temp C: 19.19  
stack V: 1.53  
stack I: 0.48  
got data:  
temp C: 19.25  
stack V: 1.50  
stack I: 0.55
```

This concludes the project, and shows the Developer Kit working to provide remote power to a wireless sensor module.

Expansion ideas

- Add more sensors,
- Send a warning when the stack voltage is getting low,
- Add an actuator on the sender that can be activated by the receiver.