Practicum 2 Betrouwbare communicatie

Peter van Dijk & Elizabeth Schermerhorn

1 juli 2014

Inhoudsopgave

1 Packet Error Rate-metingen

1.1 Inleiding

De Nordic RF24 is een radiozender en -ontvanger voor de 2.4GHz band. De RF24 heeft echter een aantal verschillende instellingen voor de kanalen, outputpower en datatransmissiesnelheid. Het doel van dit onderzoek is bepalen wat de optimale instellingen zijn om de Packet Error Rate (PER) zo laag mogelijk te krijgen. Door middel van de volgende vergelijking wordt de PER berekend:

$$\frac{[aantal\ incorrect\ ontvangen\ pakketten]}{[aantal\ correct\ ontvangen\ pakketten]}$$

Eerst zal de probleemstelling besproken worden samen met vastgestelde hypothesen, vervolgens zal de methodologie aan bod komen en als laatste zullen de resultaten besproken en geanalyseerd worden.

1.2 Probleemstelling

De onderzoeksvraag die hier beantwoord zal worden is: *"Voor welke omgevingsparameters en instellingen van de radio zal de betrouwbaarheid van de communicatie het beste zijn."* Hierbij omvatten de instellingen de outputpower, datatransmissiesnelheid en het frequentiekanaal en zal de betrouwbaarheid worden uitgedrukt met behulp van de PER. We werken met de volgende hypothesen:

1. Bij een hogere outputpower zal het pakketverlies kleiner zijn.
2. Bij een hogere datatransmissiesnelheid zal er meer pakketverlies optreden.

In dit onderzoek zijn deze hypothesen getoetst.

1.3 Methodologie

Om deze hypothesen te toetsen moeten er metingen gedaan worden. Hier zijn verschillende zaken van belang.

- De gebruikte hardware
- De gebruikte software
- De instellingen/vastgestelde constanten
- De onderzoeksopstelling

1.3.1 De hardware

De hardware die gebruikt is bij de metingen is een Arduino UNO en een Nordic nRF24L01 draadloze transceiver.

1.3.2 De software

Om de RF24 te gebruiken wordt gebruik gemaakt van de opensourcesoftware-bibliotheek voor de Arduino [2]. Met deze software kan de radio aangestuurd worden en kunnen pakketjes worden verzonden en ontvangen. De radio luistert en verstuurt pakketten over een bepaald frequentiekanaal. Het is van belang dat de radio's hetzelfde frequentiekanaal gebruiken, zodat ze met elkaar kunnen communiceren. De kanalen hebben een nummer van 0 tot en met 125. Naast het frequentiekanaal kan ook de outputpower, de datatransmissiesnelheid en de payload worden aangepast. De standaardinstellingen zijn als volgt:

- frequentiekanaal: 76
- outputpower: MAX
- datatransmissiesnelheid: 1MBps
- payload: 32 bytes

Hierbij worden de waardes van de outputpower als volgt gedefiniëerd:

- MAX: 0dBm
- LOW: -12dBm
- MIN: -18dBm

Om de metingen uit te voeren en onze hypotheses te testen is er gebruik gemaakt van één zender en één ontvanger. De zender gebruikt `sender.ino` en de ontvanger gebruikt `receiver.ino`. Deze programma's zijn te vinden in de appendix. De zender zend pakketjes met daarin het pakketnummer. De ontvanger begint met luisteren en onthoudt het nummer van het eerste pakketje dat hij heeft ontvangen. Vervolgens zal de ontvanger blijven luisteren totdat deze 1000 pakketten heeft ontvangen. Uit de inhoud van het laatste pakket kan worden opgemaakt hoeveel pakketten er verloren zijn gegaan voor de duizend die er ontvangen zijn. De PER wordt berekend met de formule:

Hierbij is [*aantal ontvangen pakketten*] dus 1000, omdat er altijd wordt gewacht tot er 1000 pakketten ontvangen zijn.

$$\frac{[laatste\ pakketnummer] - [eerste\ pakketnummer]}{[aantal\ ontvangen\ pakketten]}$$

1.3.3 De instellingen/vastgestelde constanten

Om deze metingen bruikbaar te houden moeten er een paar constanten gedefinieerd worden.

- De grootte van de payload
- Het aantal te versturen berichten

Voor de grootte van de payload is er gekozen voor de grootst mogelijke waarde die deze kan aannemen, namelijk 255 bytes. Deze waarde is gekozen omdat de onderzoeksvraag is wanneer is de radiocommunicatie het meest betrouwbaar en dit moet getest worden voor reële waarden. Het aantal te versturen berichten is op 1000 gezet. Dit aantal is groot genoeg om een consistent resultaat te verkrijgen. Hierbij zijn herzendingen van onontvangen pakketten niet meegeteld, aangezien deze niet gebruikt worden. Herzendingen zouden immers de PER veranderen, aangezien er niet gedetecteerd kan worden of een pakket de eerste keer goed is ontvangen of na een herzending.

De meetopstelling bestaat uit twee radio's welke 270cm uit elkaar geplaatst zijn. Om de resultaten consistent te houden moet bij elke meting deze afstand aangehouden worden. Verder worden bericht niet opnieuw verzonden als er geen acknowledgement volgt, want dit betekent pakketverlies en dat moet gemeten worden.

1.4 Resultaten en analyse

Hier zullen de resultaten weergegeven en geanalyseerd worden.
De constanten zoals ze standaard gebruikt zullen worden. Als ze gewijzigd worden voor de test dan wordt dit aangegeven.

- afstand tussen radio's: 270 cm
- geen hertransmissie van pakketten
- payload: 255 bytes
- frequentiekanaal: 125

1.4.1 Metingen en resultaten

Er zijn een aantal tests uitgevoerd in een ruimte waar slechts interferentie was van Wi-Fi, wat dezelfde frequentieband gebruikt (2.4GHz tot 2.5GHz [1]). De resultaten van deze tests zijn te zien in Tabel 1. Verder zijn er nog enkele kleinere tests uitgevoerd op een netwerk met niet alleen Wi-Fi, maar ook andere vergelijkbare testopstellingen, wat voor extra interferentie zorgt. De resultaten van de tests met veel interferentie zijn te zien in Tabel 2.

Om het resultaat nauwkeuriger en accurater te kunnen weergeven is ervoor gekozen om alle tests in drievoud uit te voeren zodat vreemde gebeurtenissen opgemerkt worden. Een voorbeeld hiervan is een error rate groter dan 1. Tussen deze resultaten is de instelling -6dBm voor outputpower niet weergegeven, aangezien deze voor de gebruikte versie van de radio niet een mogelijke waarde was.

Wat opvalt aan de meetresultaten is dat de PER bij een outputpower van MIN aanzienlijk hoger is dan bij de rest van de tests. Verder blijkt ook de hoeveelheid interferentie een grote rol te spelen.

Tabel 1: PER metingen

Instellingen	PER	gemiddelde PER
outputpower: MAX datatransmissiesnelheid: 2MBps delay: 25 ms	1/1000 1/1000 0/1000	2/3000
outputpower: MAX datatransmissiesnelheid: 2MBps delay: 50 ms	0/1000 0/1000 0/1000	0/3000
outputpower: MAX datatransmissiesnelheid: 1MBps delay: 50 ms	0/1000 0/1000 0/1000	0/3000
outputpower: MIN datatransmissiesnelheid: 2MBps delay: 50 ms	85/1000 669/1000 519/1000	1273/3000
outputpower: LOW datatransmissiesnelheid: 2MBps delay: 50 ms	0/1000 2/1000 6/1000	8/3000
outputpower: MAX datatransmissiesnelheid: 2MBps delay: 50 ms channel: 0	0/1000 0/1000 0/1000	0/3000
outputpower: MAX datatransmissiesnelheid: 2MBps delay: 50 ms channel: 64	0/1000 1/1000 0/1000	1/3000

Tabel 2: PER metingen met veel interferentie

Instellingen	PER	gemiddelde PER
outputpower: MAX	47/300	433/3000
datatransmissiesnelheid: 2MBps	77/300	
delay: 50 ms	9/300	

1.5 Conclusie

Als er naar de gevonden waarden gekeken wordt, is de PER zeer hoog wanneer de outputpower op MIN staat. Een gemiddelde PER van 1273/3000 pakketten zorgt op den duur voor veel hertransmissies, dus veel vertraging op het netwerk. Dit is een situatie die niet gewenst is. Wanneer naar de overige instellingen gekeken wordt en de daarbij behorende gemiddelde PER dan zijn ze perfect of een PER kleiner dan 0.01. Hier moet bij vermeld worden dat op het moment dat er meer interferentie is op het netwerk de PER toeneemt. Op dat moment is het kiezen van een ongebruikt frequentiekanaal van belang.

De onderzoeksvraag is: *"Voor welke waarden voor de outputpower, datatransmissiesnelheid en frequentiekanaal is de PER het laagst."* Het antwoord hierop is dat voor elke waarde van de datatransmissiesnelheid en het frequentiekanaal de PER niet boven de 0.01 uit komt op een afstand van 270cm. Dit geldt echter niet voor de outputpower; op het moment dat de outputpower op MIN gezet wordt bedraagt de gemiddelde PER 1273/3000. Hieruit kan geconcludeerd worden dat de outputpower tenminste op LOW of MAX gezet moet worden om de PER tot een minimum te beperken.

2 Betrouwbare end-to-end-communicatie

2.1 Inleiding

In het vorige experiment is de betrouwbaarheid van een singlehop-netwerk onderzocht. Nu kan de betrouwbaarheid van een multihop-netwerk getoetst worden en kunnen de waarden vergeleken worden om te bepalen welk netwerk betrouwbaarder is.

2.2 Probleemstelling

De probleemstelling waarvan uit zal worden gegaan is *"Is het mogelijk een betrouwbare end-to-end multihop-connectie op te zetten?"*. De hypothese is dat dit mogelijk is in combinatie met :

- Het gebruik van hertransmissies
- Het gebruik van routingtabellen

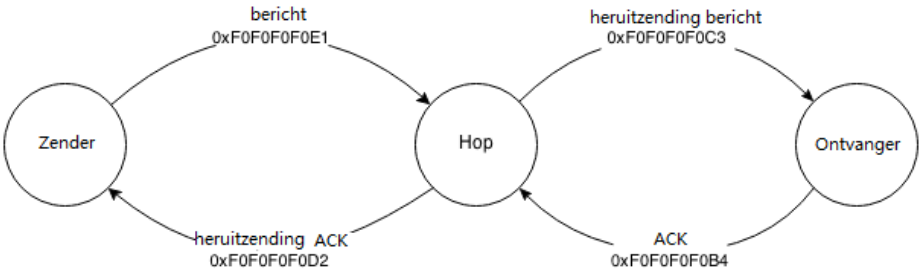
Met deze aspecten verwerkt in de implementatie zou het multihop-netwerk moeten werken.

2.3 Protocol

Het gebruikte protocol omvat de volgende zaken:

- gebruik van routing tabel
- zenden van hertransmissies

Het protocol zorgt ervoor dat wanneer er geen acknowledgement ontvangen wordt er een hertransmissie gestuurd wordt. Deze functionaliteit wordt geleverd door de RF24 opensourcesoftwarebibliotheek [2]. Ter ondersteuning van deze hertransmissiefunctionaliteit worden er automatisch ACKs gezonden om te bevestigen dat een pakket correct ontvangen is. *Deze ACKs en hertransmissies vinden alleen plaats per hop en niet tussen eindnodes.* Hier is voor gekozen omdat de PER moet worden gemeten van de verbinding tussen zender en ontvanger, waarbij hertransmissies de PER niet zouden veranderen, aangezien we geen onderscheid maken in de telling tussen het zenden van een nieuw pakket en het herzenden van een oud pakket.



Figuur 1: Opstelling van het netwerk

2.3.1 Het netwerk

De gebruiktee netwerkopstelling bestaat uit drie nodes: een zender (`Sender`), een hopnode (`Hop`) en een ontvanger (`Receiver`). De zender zendt een bericht naar de hopnode, de hopnode zendt dit bericht naar de ontvanger, de ontvanger zendt een ACK (ontvangstbevestiging) naar de hopnode en de hopnode stuurt deze ACK door naar de zender. In de implementatie komt het protocol terug in alle nodes. Voor iedere verbinding waarover berichten verstuurd worden is er hertransmissiefunctionaliteit. Verder heeft de hopnode een grotere routing tabel, aangezien deze berichten moet doorsturen van zender naar ontvanger en andersom.

2.4 Methodologie

Om de onderzoeksvraag te kunnen beantwoorden is er een meetopstelling nodig. De meetopstelling is weergegeven in Figuur 1 met een zendernode, een hopnode en een ontvangernode. De namen geven het al aan, de zender stuurt de berichten (en wacht op bevestiging), de hopnode stuurt de berichten door en de ontvanger ontvangt de berichten (en bevestigd de ontvangst).

2.4.1 De software

De software die gebruikt is voor dit experiment is voor de `Sender` en `Receiver` grotendeels hetzelfde als het voorbeeld pingpair dat wordt meegeleverd bij de RF24 library voor de Arduino [2]. Voor de implementatie is slechts de ondersteuning voor meerdere adressen van de verbindingen ingebouwd en is de pakketgrootte veranderd naar 255 bytes. Deze addressen zijn te zien in Figuur 1. De zender heeft een uitgaande verbinding op adres 0xF0F0F0E1 en een inkomende verbinding op adres 0xF0F0F0D2 naar de hopnode. De ontvanger heeft een inkomende verbinding op adres 0xF0F0F0C3 en een uitgaande verbinding op adres 0xF0F0F0B4 naar de hopnode.

De `Hop` luistert naar de zender en ontvanger en stuurt berichten van de zender door naar de ontvanger en andersom. Hierbij wordt gebruik gemaakt van de eerder

De RF24 opensourcesoftwarebibliotheek handelt de communicatie verder af en verzorgt de nodige hertransmissies. Voor de instelling van de radioverbinding is gebruik gemaakt van een aantal constanten, namelijk:

- payload: 255 bytes
- hertransmissies : 15
- delay: 50ms
- datatransmissiesnelheid: 2MBps

Deze constanten zullen tijdens het testen niet worden veranderd.

2.4.2 De hardware

Deze bestaat uit drie identieke nodes. Deze nodes zijn allemaal voorzien van een radio waarmee ze de berichten kunnen ontvangen en versturen. De nodes zijn op één lijn geplaatst met 100 cm afstand tussen zender en hopnode en tussen hopnode en ontvanger. De experimenten zijn uitgevoerd in een netwerk waar nog meer radioś aan het sturen zijn.

2.5 Resultaten en analyse

Om de resultaten te verkrijgen die gewenst zijn worden er telkens andere waarden gebruikt voor de transmissie sterkte. De outputpower geldt voor alle drie de nodes(zender, hop en ontvanger). De drie waarden die hiervoor gebruikt zijn MIN, LOW en MAX, zoals besproken in sectie 1.3.2. De waarden die gevonden zijn bij de instellingen zijn te zien in Tabel 3. Hierin staan de PER samen met de outputpower waarmee de pakketjes verstuurd zijn. Het opvallende aan deze resultaten is dat de PER voor de outputpower MIN ook zeer laag is in vergelijking met de PER die in sectie 1.4.1 gegeven is.

Tabel 3: PER metingen

Instellingen	PER	gemiddelde PER
outputpower: MIN	1/1000 17/1000 7/1000	25/3000
outputpower: LOW	0/1000 1/1000 1/1000	2/3000
outputpower: MAX	0/1000 0/1000 1/1000	1/3000

2.6 Conclusie

De hypothese die gesteld werd: *"Is het mogelijk een betrouwbaar end-to-end multihop-connectie op te zetten?"*. Deze hypothese wordt nu gestaafd door de resultaten die verkregen zijn door de experimenten uit te voeren met veranderende outputpowers. Uit de resultaten kan opgemaakt worden dat een multi-hop verbinding, die op deze manier geïmplementeerd is, een redelijke implementatie geeft met een error rate kleiner dan 1%. Zoals verwacht in de hypothese is zenden met outputpower op MIN de slechtst scorende instelling, maar de PER is nog zeer laag. Dit zou kunnen komen door het gebruik van hertransmissies op de tussenverbindingen.

Referenties

[1] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-speed physical layer extension in the 2.4 GHz band. *IEEE Std. 802.11b*, 1999.

[2] RF24 Class Reference. `http://maniacbug.github.io/RF24/classRF24.html`, May 2014.

A De radio en de RF24-library

A.1 Eigenschappen van de RF24

In dit onderzoek wordt gebruik gemaakt van de rf24 radio. Deze radio heeft de volgende eigenschappen:

- **Frequentieband:** 2.4000-2.4835 GHz
- **Datatransmissiesnelheid:** 1 of 2 Mb/s
- **Aantal kanalen:** 126 RF-kanalen
- **Modulatietechniek:** Gaussian Frequency Shift Key(GFSK)

A.2 Energieverbruik

De radio heeft het volgende energieverbruik in de verschillende modi:

Energiemodus	**Energieverbruik in Ampère**
Standby-I	22 μA
Standby-II	320 μA
Power down	900 nA

Zendmodus	**Energieverbruik in Ampère**
0 dBm	11.3 mA
-6 dBm	9 mA
-12 dBm	7.5 mA
-18 dBm	7 mA

Ontvangstmodus	**Energieverbruik in Ampère**
2 Mbps	12.3 mA
1Mbps	11.8 mA

A.3 Enhanced ShockBurst

Een belangrijk aspect van de RF24 radio is Enhanced ShockBurst. Dit is een pakketgerichte datalinklayer protocol. Belangrijke functies van Enhanched ShockBurst zijn:

- Automatische pakket samenvoeging
- 6 data pipe multiceivers
- pakket afhandeling

Het formaat van het datapakket bestaat uit verschillende delen, het adres, pakket identificatie, geen acknowledgement bits en CRC. De velden hebben de volgende functies:

Data veld	Functie
Adres	Het adres van de ontvanger
Pakket identificatie	Om te achterhalen of het pakket een hertransmissie is.
geen acknowledgement bit	Als deze bit gezet is mag het pakket niet automatisch erkend worden.
CRC	Error detectie mechanisme voor het pakket

A.3.1 Automatische hertransmissie

Enhanced Shockburst heeft als eigenschap automatische pakketvalidatie. In ontvangstmodus zoekt de RF24 constant naar valide adressen welke gegeven zijn in RX_ADDR registers. Op het moment dat een valide adres gedetecteerd wordt, zal Enhanced Shockburst het pakket gelijk valideren. Enhanced Shockburst is voorzien van een ART(Automatic Retransmission) functie. Door deze functie is Enhanced Shockburst in staat een pakket uit zichzelf opnieuw te versturen als er geen acknowledgement wordt ontvangen. Er zijn drie redenen waarom de ART functie stopt met het luisteren na het versturen van een pakket. De ART stopt met luisteren na:

- Auto retransmit delay(ARD) is verlopen
- Geen gematcht adres gevonden binnen 250μ s
- Nadat er een pakket ontvangen is

De radio is niet in staat om te broadcasten maar om deze functionaliteit te gebruiken moet er gebruik gemaakt worden van een multiceiver.

A.4 Adressering

De adressen van een multiceiver moeten altijd sterk op elkaar lijken, alleen de least significant byte moet anders zijn. De volgende 4 adressen volgen uit dit principe:

- F0F0A
- F0F0B
- F0F0C
- F0F0D

Doordat de Least significant byte anders is zijn dit vier geldige adressen.

A.5 Functies

Met de volgende functies worden waarden van de RF24 radio ingesteld:

Functie	Methode
Frequentiekanaal instellen	void RF24::setChannel (uint8_t channel)
Zendvermogen	void setPALevel (rf24_pa_dbm_e level)
Aantal pogingen van ART instellen	void RF24::setRetries (uint8_t delay, uint8_t count)
Adres van de zender instellen	void openWritingPipe (uint64_t address)
Adres van de ontvanger instellen	void openReadingPipe (uint8_t number, uint64_t address)

B Code Sender

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h"

RF24 radio(3, 9);
int messages_sent = 0;

const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };

void setup(void)
{
  Serial.begin(57600);
  printf_begin();
  printf("\n\rRF24/examples/pingpair/\n\r");
  //
  // Setup and configure rf radio
  //
  radio.begin();
  radio.setPALevel(RF24_PA_MAX);
```

```
  radio.setDataRate(RF24_2MBPS);
  radio.setRetries(0,0);
  radio.setPayloadSize(255);
  radio.setChannel(125);
  radio.openWritingPipe(pipes[0]);
  radio.startListening();
  radio.printDetails();
}
void loop(void){
    // First, stop listening so we can talk.
    radio.stopListening();

    printf("Now sending...");
    bool ok = radio.write( &messages_sent, sizeof(unsigned int) );
    messages_sent++;
    printf("number of messages sent: %u\n\r",messages_sent);

    // Now, continue listening
    radio.startListening();

// Wait before sending next message
    delay(25);
}
```

C Code Receiver

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h"

RF24 radio(3, 9);

const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };

unsigned long received = 0;
unsigned long sent = -1;
unsigned long initVal = -1;
bool started = false;
bool stopped = false;

void setup(void)
{
```

```
  Serial.begin(57600);
  printf_begin();
  received = 0;
  printf("\n\rRF24/examples/pingpair/\n\r");
  //
  // Setup and configure rf radio
  //
  radio.begin();
  radio.printDetails();
  radio.printDetails();
  radio.setChannel(125);
  radio.setRetries(0,0);
  radio.printDetails();
  radio.setRetries(15,15);

  Serial.print();
  Serial.print(radio.getPALevel());
  radio.setPayloadSize(255);
  radio.setDataRate(RF24_2MBPS);
  radio.openReadingPipe(1,pipes[0]);
  radio.startListening();
  radio.printDetails();
}

void loop(void)
{
  if ( radio.available() )
  {
    bool done = false;
    while (!done)
    {
      done = radio.read( &sent, sizeof(unsigned long) );
      stopped = received >= 1000;
      if(started && !stopped)
      {
        received = received+1;
        printf("Got packet: %u", received);
        printf(" payload: %u", (sent-initVal));
        printf(" lost: %u\r\n", ((sent-initVal)-received));
      }
      else if(!started && sent != -1 )
      {
        started = true;
        initVal = sent;
      }
      // Delay just a little bit to let the other unit
```

```
      // make the transition to receiver
      delay(20);
    }
    // First, stop listening so we can talk
    radio.stopListening();

    // Now, resume listening so we catch the next packets.
    radio.startListening();
  }
}
```

D Code Hop

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h"

RF24 radio(3, 9);
int messages_sent = 0;

const uint64_t pipes[4] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL, 0xF0F0F0F0C3LL, 0xF0F0F0F0B4LL};
const uint8_t readingPipeSender = 1;
const uint8_t readingPipeReceiver = 2;
uint8_t receivedFrom;

bool listen_receiver = true;

void setup(void)
{
  Serial.begin(57600);
  printf_begin();
  printf("\n\rRF24/examples/pingpair/\n\r");
  radio.begin();
  radio.setRetries(15,15);
  radio.setChannel(48);
  radio.setPayloadSize(255);
  radio.setPALevel(RF24_PA_MIN);
  radio.setDataRate(RF24_2MBPS);
  radio.openReadingPipe(readingPipeSender,pipes[0]);
  radio.openReadingPipe(readingPipeReceiver,pipes[3]);
  //
  // Start listening
```

```
  //
  radio.startListening();
  radio.printDetails();
}

void loop(void){
  // Message from sender
  if ( radio.available( &receivedFrom ) )
  {
    if(receivedFrom == readingPipeSender)
    {
      unsigned long message;
      bool done = false;
      while (!done)
      {
        done = radio.read( &message, sizeof(unsigned long) );
        printf("Got payload %lu...",message);
        // Delay just a little bit to let the other unit
        // make the transition to receiver
        delay(5);
      }
      // First, stop listening so we can talk
      radio.stopListening();
      radio.openWritingPipe(pipes[2]);
      radio.write( &message, sizeof(unsigned long) );
      printf("Relayed message to receiver.\n\r");
      // Now, resume listening so we catch the next packets.
      radio.startListening();
    }
    //Message from receiver
    else if(receivedFrom == readingPipeReceiver)
    {
      unsigned long message;
      bool done = false;
      while (!done)
      {
        done = radio.read( &message, sizeof(unsigned long) );
        printf("Got payload %lu...",message);
        // Delay just a little bit to let the other unit
        // make the transition to receiver
        delay(5);
      }
      // First, stop listening so we can talk
      radio.stopListening();
      radio.openWritingPipe(pipes[1]);
      // Send the final one back.
```

```
      radio.write( &message, sizeof(unsigned long) );
      printf("Relayed message to sender.\n\r");
      // Now, resume listening so we catch the next packets.
      radio.startListening();
    }
  }
}
```