Ostafichuk Blog

MODBUS on the Pi Part 1, compiling a basic example

I have started using the Pi with the Raspbian distribution directly from the raspberrypi.org website. I will not go into the steps to set up your pi for the first time because so many others have done a way better job than I could ever do.
You can run my examples with or without the GUI installed. The purpose of this project is to monitor things around the house, so you do not need the GUI, and I recommend running just the command line.

The first thing you need to do is log onto your pi using ssh (or putty in windows).
$**ssh pi@192.168.1.?** (get the IP address from your network router status page)

Next, make a directory to work in, and cd into it
$**mkdir modbus**
$**cd modbus**

Now we create the file to hold the c code by using the nano text editor.
$**nano bandwidth-server-one.c**

Paste the following code into the file. (I got this from the test files for modbus at libmodbus.org) I have also included a link to the file so you can download it and save it on your pi. bandwidth-server-one

```
/*
 * Copyright © 2008-2010 Stéphane Raimbault
 *
 * This program is free software: you can redistribute it and/or modify
```

```
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see .
 */

#include
#include
#include
#include
#include

#include

enum {
    TCP,
    RTU
};

int main(int argc, char *argv[])
{
    int socket;
    modbus_t *ctx;
    modbus_mapping_t *mb_mapping;
    int rc;
    int use_backend;
    int nPort = 1502;

     /* TCP */
    if (argc > 1) {
        if (strcmp(argv[1], "tcp") == 0) {
            use_backend = TCP;
        } else if (strcmp(argv[1], "rtu") == 0) {
            use_backend = RTU;
        } else {
            printf("Usage:\n  %s [tcp|rtu] - Modbus client to measure data
bandwith\n\n", argv[0]);
            exit(1);
        }
```

```
    } else {
        /* By default */
        use_backend = TCP;
        printf("bandwidth-server-one:\n Running in tcp mode - Modbus client
to measure data bandwith\n\n");
    }

    if (use_backend == TCP) {
        printf("Waiting for TCP connection on Port %i \n",nPort);
        ctx = modbus_new_tcp("127.0.0.1", nPort);
        socket = modbus_tcp_listen(ctx, 1);
        modbus_tcp_accept(ctx, &socket);
        printf("TCP connection started!\n");
    } else {
        printf("Waiting for Serial connection on /dev/ttyUSB0\n");
        ctx = modbus_new_rtu("/dev/ttyUSB0", 115200, 'N', 8, 1);
        modbus_set_slave(ctx, 1);
        modbus_connect(ctx);
        printf("Serial connection started!\n");
    }

    mb_mapping = modbus_mapping_new(MODBUS_MAX_READ_BITS, 0,
                                    MODBUS_MAX_READ_REGISTERS, 0);
    if (mb_mapping == NULL) {
        fprintf(stderr, "Failed to allocate the mapping: %s\n",
                modbus_strerror(errno));
        modbus_free(ctx);
        return -1;
    }

    for(;;) {
        uint8_t query[MODBUS_TCP_MAX_ADU_LENGTH];

        rc = modbus_receive(ctx, query);
        if (rc >= 0) {
            printf("Replying to request.\n");
            modbus_reply(ctx, query, rc, mb_mapping);
        } else {
            /* Connection closed by the client or server */
            break;
        }
    }

    printf("Quit the loop: %s\n", modbus_strerror(errno));

    modbus_mapping_free(mb_mapping);
```

```
    close(socket);
    modbus_free(ctx);

    return 0;
}
```

Ctrl-X to exit, hit **Y** to save

Next we need to install the libmodbus packages
$**sudo apt-get install libmodbus5 libmodbus-dev**

Finally, we want to compile the code and create the binary executable
$**gcc bandwidth-server-one.c -o bandwidth-server-one `pkg-config –libs –cflags libmodbus`**

If the above command does not work, try the following
$gcc -I /usr/include/modbus bandwidth-server-one.c -o bandwidth-server-one -L/usr/lib/modbus -lmodbus

To run the code type
$./bandwidth-server-one

If everything worked properly, you should see the output:

```
bandwidth-server-one:
Running in tcp mode - Modbus client to measure data bandwith


 Waiting for TCP connection on Port 1502
```

Congratulations! You are now running a simple modbus server on the Raspberry Pi. The current example does not do much of interest, other than it is used as a bandwidth tester, but later we will be changing the program to be able to read and

control the GPIO pins on the Pi, then it will get interesting!
Hit **Ctrl-C** to exit the program.

Continue to MODBUS on the Pi Part 2, Adding functionality for testing