Peidong Qi
05/01/2018
AMRUPT, Spring 2018

RTL SDR to Raspberry Pi Connection and Datalogging

Goals

Adding custom blocks on Raspberry Pi

General approach

For this week, we still work on the GRC, Russell has made progress on running grc on PC, we already can re-create direction-finding project flowgraph on PC. However, we have a problem on running grc on Raspberry Pi. There are several tools can help us to install custom blocks for GRC on Raspberry pi. Last week, we have tried “gr-modtool”. This tool is most original block creating tool. This week, we have found a new tool called “python boobms”, this tools can install custom block very easily, but you need to upload your block source code to their cloud library. We also have a trouble when we try to install this software. Our Raspberry pi has crashed due to memory runout. We were not able to install python boobms on Raspberry pi. Then we keep using “gr-modtool” for our project. When we use gr-modtool to add custom block, the configure error keeps showing up. The Raspberry pi keep showing we have a missing package. I will keep working that until we fixed it

Custom Block coding

- Coding Structure
    - Public Header Files

```
#ifndef INCLUDED_FOO_BAR_H
#define INCLUDED_FOO_BAR_H

#include <foo/api.h>
#include <gr_sync_block.h>

namespace gr {
  namespace foo {

    class FOO_API bar : virtual public gr_sync_block
    {
    public:

      // gr::foo::bar::sptr
      typedef boost::shared_ptr sptr;

      /*!
       * \class bar
       * \brief A brief description of what foo::bar does
       *
       * \ingroup _blk
       *
       * A more detailed description of the block.
       *
       * \param var explanation of argument var.
       */
      static sptr make(dtype var);

      virtual void set_var(dtype var) = 0;
      virtual dtype var() = 0;
    };

  } /* namespace foo */
} /* namespace gr */

#endif /* INCLUDED_FOO_BAR_H */
```

■ Implementation Header File

```
#ifndef INCLUDED_FOO_BAR_IMPL_H
#define INCLUDED_FOO_BAR_IMPL_H

#include <foo/bar.h>

namespace gr {
  namespace foo {

    class FOO_API bar_impl : public bar
    {
    private:
      dtype d_var;

    public:
      bar_impl(dtype var);

      ~bar_impl();

      void set_var(dtype var);
      dtype var();

      int work(int noutput_items,
          gr_vector_const_void_star &input_items,
          gr_vector_void_star &output_items);
    };

  } /* namespace foo */
} /* namespace gr */

#endif /* INCLUDED_FOO_BAR_IMPL_H */
```

- Implementation Source File

```
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include "bar_impl.h"
#include <gr_io_signature.h>

namespace gr {
  namespace foo {

    bar::sptr bar::make(dtype var)
    {
      return gnuradio::get_initial_sptr(new bar_impl(var));
    }

    bar_impl::bar_impl(dtype var)
      : gr_sync_block("bar",
              gr_make_io_signature(1, 1, sizeof(in_type)),
              gr_make_io_signature(1, 1, sizeof(out_type)))
    {
      set_var(var);
    }

    bar_impl::~bar_impl()
    {
      // any cleanup code here
    }

    dtype
    bar_impl::var()
    {
      return d_var;
    }

    void
    bar_impl::set_var(dtype var)
    {
      d_var = var;
    }

    int
    bar_impl::work(int noutput_items,
                   gr_vector_const_void_star &input_items,
                   gr_vector_void_star &output_items)
    {
      const in_type *in = (const in_type*)input_items[0];
      out_type *out = (out_type*)output_items[0];

      // Perform work; read from in, write to out.

      return noutput_items;
    }

  } /* namespace foo */
} /* namespace gr */
```

■ SWIG Interface File

```
#define FOO_API

%include "gnuradio.i"

//load generated python docstrings
%include "foo_swig_doc.i"

%{
#include "foo/bar.h"
%}

%include "foo/bar.h"

GR_SWIG_BLOCK_MAGIC2(foo, bar);
```

- Block Structure

- The work function

```
int work(int noutput_items,
         gr_vector_const_void_star &input_items,
         gr_vector_void_star &output_items)
{
  //cast buffers
  const float* in0 = reinterpret_cast(input_items[0]);
  const float* in1 = reinterpret_cast(input_items[1]);
  float* out = reinterpret_cast(output_items[0]);

  //process data
  for(size_t i = 0; i < noutput_items; i++) {
    out[i] = in0[i] + in1[i];
  }

  //return produced
  return noutput_items;
}
```

- IO signatures

```
-- A block with 2 inputs and 1 output --

gr_sync_block("my adder", gr_make_io_signature(2, 2, sizeof(float)), gr_make_io_signature(1, 1, sizeof(float)))

-- A block with no inputs and 1 output --

gr_sync_block("my source", gr_make_io_signature(0, 0, 0), gr_make_io_signature(1, 1, sizeof(float)))

-- A block with 2 inputs (float and double) and 1 output --

std::vector input_sizes;
input_sizes.push_back(sizeof(float));
input_sizes.push_back(sizeof(double));

gr_sync_block("my block", gr_make_io_signaturev(2, 2, input_sizes), gr_make_io_signature(1, 1, sizeof(float)))
```

- Block types

- Synchronous Block
- Decimation Block
- Interpolation Block
- Basic Block
- Other Types of Blocks

To build a custom block, we need to edit the source code by following the instruction above, then we need to run gr-modtool to add those blocks on grc. We have already figure out the source code for each custom blocks in direction finding project. And we are able to run test for that.

Planned Course of Action

Run the test for direction-finding project from coherence-receiver website. I also need to find out how to solve configure issue on Raspberry Pi.

Resources and relevant Forum Posts

[1] "RTL-SDR Blog silver dongle first impressions, compared to NooElec blue dongle"https://medium.com/@rxseger/rtl-sdr-blog-silver-dongle-first-impressions-compared-to-nooelec-blue-dongle-4053729ab8c7
[2] "VIDEO TUTORIAL: INSTALLING GQRX AND RTL-SDR ON A RASPBERRY PI"https://www.rtl-sdr.com/video-tutorial-installing-gqrx-and-rtl-sdr-on-a-raspberry-pi/
[3] "DIGITAL RADIO WITH A RASPBERRY PI" http://www.michaelcarden.net/?p=48
[4] "Guided Tutorial GNU Radio in Python
"https://wiki.gnuradio.org/index.php/Guided_Tutorial_GNU_Radio_in_Python
[5] "GRCon17 - Real-Time Direction Finding Using Two Antennas on an Android Phone - Sam Whiting" https://www.youtube.com/watch?v=jptYYiHth8U
[6]" Real-Time Direction Finding Using Two Antennas on an Android Phone - Sam Whiting" https://www.gnuradio.org/wp-content/uploads/2017/12/Todd-Moon-Gnuradio-DOA.pdf
[7]"gnuradio-doa" https://github.com/samwhiting/gnuradio-doa/tree/master/flowgraphs
[8]" GNURadioCompanion" https://wiki.gnuradio.org/index.php/GNURadioCompanion
[9] "Install RTL-SDR on Raspberry PI"
https://gist.github.com/floehopper/99a0c8931f9d779b0998