Automatic Meter Reading

By: Biruh Dejene

Abstract

The main goal of this mini-project is to use MATLAB Communications Toolbox to show how to read utility meters by processing Standard Consumption Message (SCM) signals and Interval Data Message (IDM) signals which are emitted by Encoder-Receiver-Transmitter (ERT) compatible meters. We can either use recorded data from a file or receive over-the-air signals in real-time using an RTL-SDR or ADALM-PLUTO radio. In order to receive signals in real-time, we used one of these SDR devices and the corresponding support package Add-On: RTL-SDR radio and the corresponding Communications Toolbox Support Package for RTL-SDR Radio.

Background

Automatic Meter Reading (AMR) is a technology that autonomously collects the consumption and status data from utility meters (such as electric, gas, or water meters) and delivers the data to utility providers for billing or analysis purposes.This technology mainly saves utility providers the expense of periodic trips to each physical location to read a meter. Another advantage is that billing can be based on near real-time consumption rather than on estimates based on past or predicted consumption

The AMR system utilizes low power radio frequency (RF) communication to transmit meter readings to a remote receiver. The RF transmission properties include:

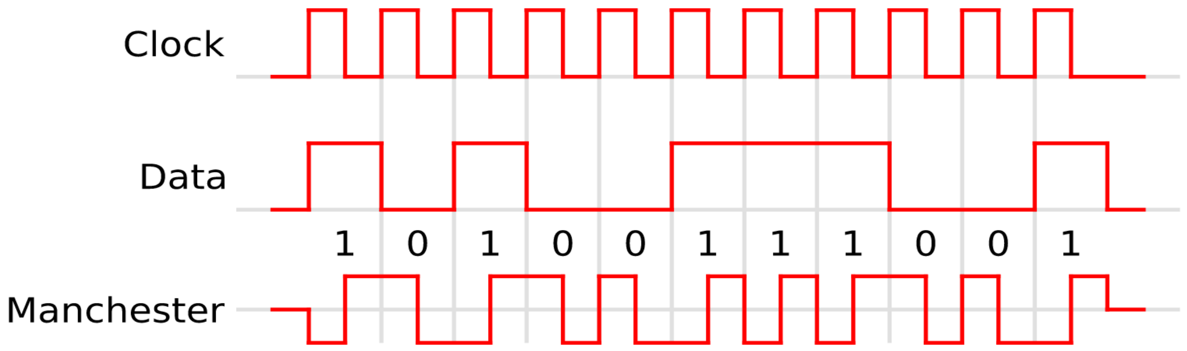
- Transmission frequency within range: 910-920 MHz
- Data rate: 32768 bps
- On-off keyed Manchester coded (also known as **phase encoding)** signaling.

Encoder Receiver Transmitters (ERT) are digital meters that utilize a low-power radio signal to communicate with hand-held receivers used by meter readers. The technology is used to transmit data from utility meters over a short-range so a utility vehicle can collect meter data without a worker physically inspecting each meter. ERT is the simplest form of amplitude-shift keying (ASK) modulation modulated radio signal. In its simplest form, the presence of a carrier for a specific duration represents a binary one, while its absence for the same duration represents a binary zero

The SCM and IDM are two types of the conventional message types that the meters send out. SCM messages are 12 bytes. Each message contains a single, cumulative meter reading value along with the meter serial number, commodity type, and checksum and tamper flags. On the other hand, IDM messages are 92 bytes and contain the time of use consumption data.

Encoding and decoding

Manchester code always has a transition at the middle of each bit period and may (depending on the information to be transmitted) have a transition at the start of the period also. The direction of the mid-bit transition indicates the data. Transitions at the period boundaries do not carry information. They exist only to place the signal in the correct state to allow the mid-bit transition.

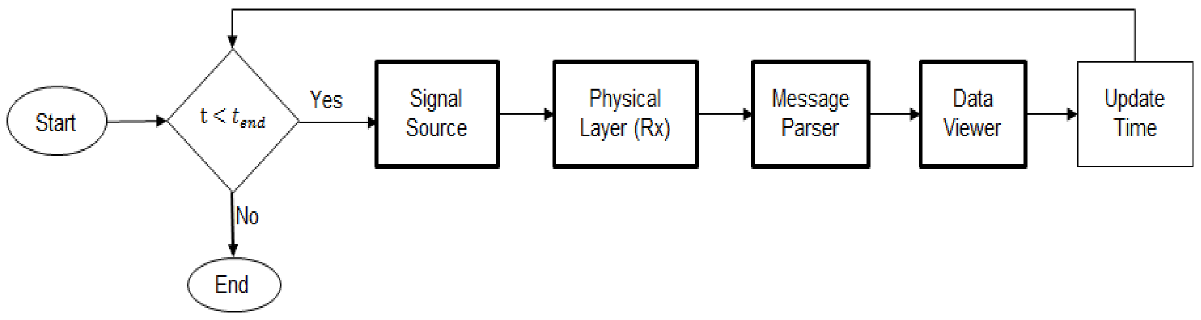


Decoding: The existence of guaranteed transitions allows the signal to be self-clocking, and also allows the receiver to align correctly; the receiver can identify if it is misaligned by half a bit period, as there will no longer always be a transition during each bit period.

Encoding: Encoding conventions for Manchester code act as an XOR.

- ❖ Each bit is transmitted in a fixed time (the "period").
- ❖ A 0 is expressed by a low-to-high transition, a 1 by a high-to-low transition
- ❖ The transitions which signify 0 or 1 occur at the midpoint of a period.
- ❖ Transitions at the start of a period are overhead and don't signify data

Receiver Code Structure



When we run the code(the code is attached below):

- The receiver initializes the simulation parameters and calculates the AMR parameters.
- A data viewer display shows the meter ID, consumption information, and commodity type
- The simulation loop calls the signal source, physical layer, message parser, and data viewer.
- The processing loop keeps track of the radio time using the frame duration.

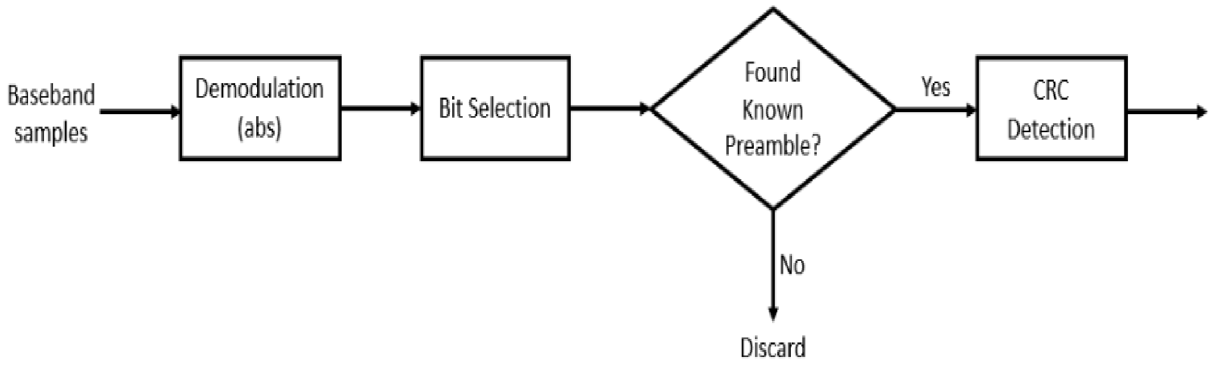
- The display updates for each data capture, showing unique meter IDs with the latest consumption information

Message Parser: For a valid message, the bits are then parsed into the specific fields of the SCM or the IDM format.

Data Viewer: The data viewer shows the decoded packets on a separate MATLAB figure. For each successfully decoded packet, the meter ID, commodity type, AMR packet type, consumption information, and the capture time are shown. As data is captured and decoded, the application lists the information decoded from these messages in a tabular form. The table lists only the unique meter IDs with their latest consumption information.

- ❖ **Meter ID** - Change the meter ID from 0, which is the default value and is reserved for displaying all detected meters, to a specific meter ID that you would like to be displayed.
- ❖ **Log data to file** - Save the decoded messages in a TXT file. You can use the saved data for post-processing**.**

Physical Layer

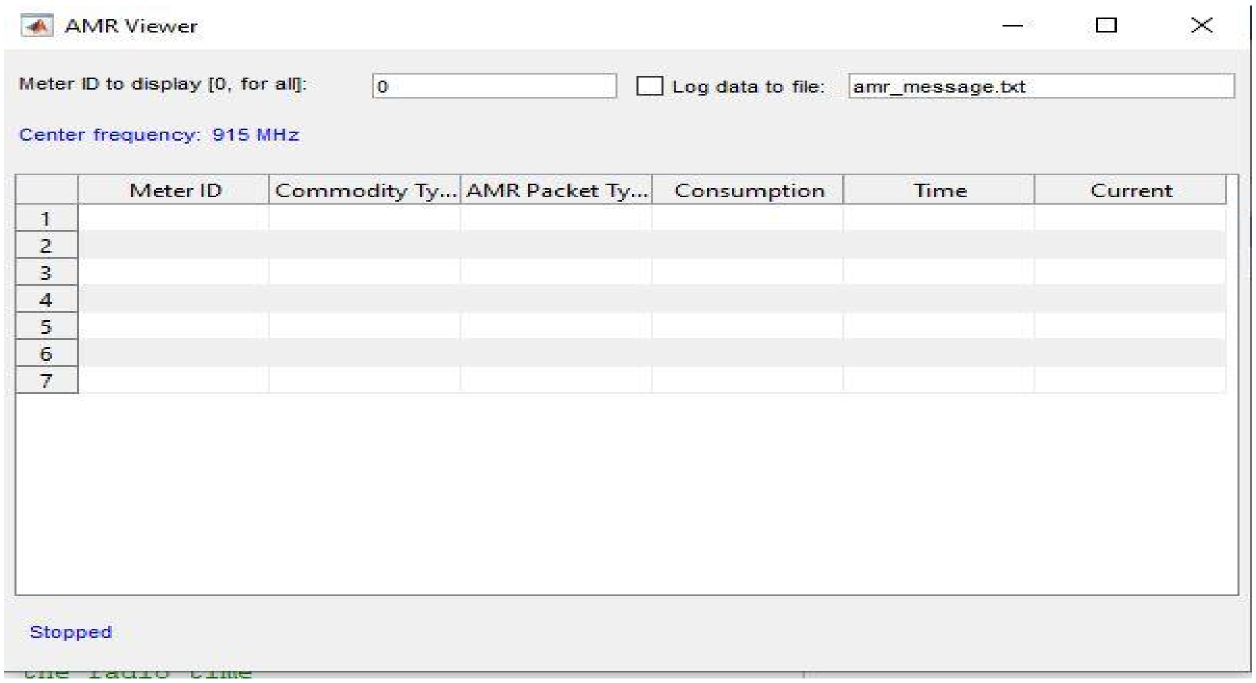


The RTL-SDR radio is capable of using a sampling rate in the range of 900-2560 kHz. A sampling rate of 1.0 Msps is used to produce a sufficient number of samples per Manchester encoded data bit. For each frequency in the hopping pattern, every AMR data packet is transmitted. The frequency hopping allows for increased reliability over time. Since every packet is transmitted on each frequency hop, it is sufficient to monitor only one frequency for this example. The radio is tuned to a center frequency of 915 MHz for the entire simulation runtime.

The received complex samples are amplitude demodulated by extracting their magnitude. The on-off keyed Manchester coding implies the bit selection block includes clock recovery. This block outputs bit sequences (ignoring the idle times in the transmission) which are subsequently checked for the known preamble. If the preamble matches, the bit sequence is further decoded, otherwise, it is discarded and the next sequence is processed.

When the known SCM preamble is found for a bit sequence, the received message bits are decoded using a shortened (255,239) BCH code which can correct up to two bit errors. In the case where the known IDM preamble is found, the receiver performs a cyclic redundancy check (CRC) of the meter serial number and of the whole packet starting at the Packet type (the 5th byte) to determine if the packet is valid. Valid, corrected messages are passed onto the AMR message parser.

Result:


AMR Viewer

Meter ID to display [0, for all]: 0

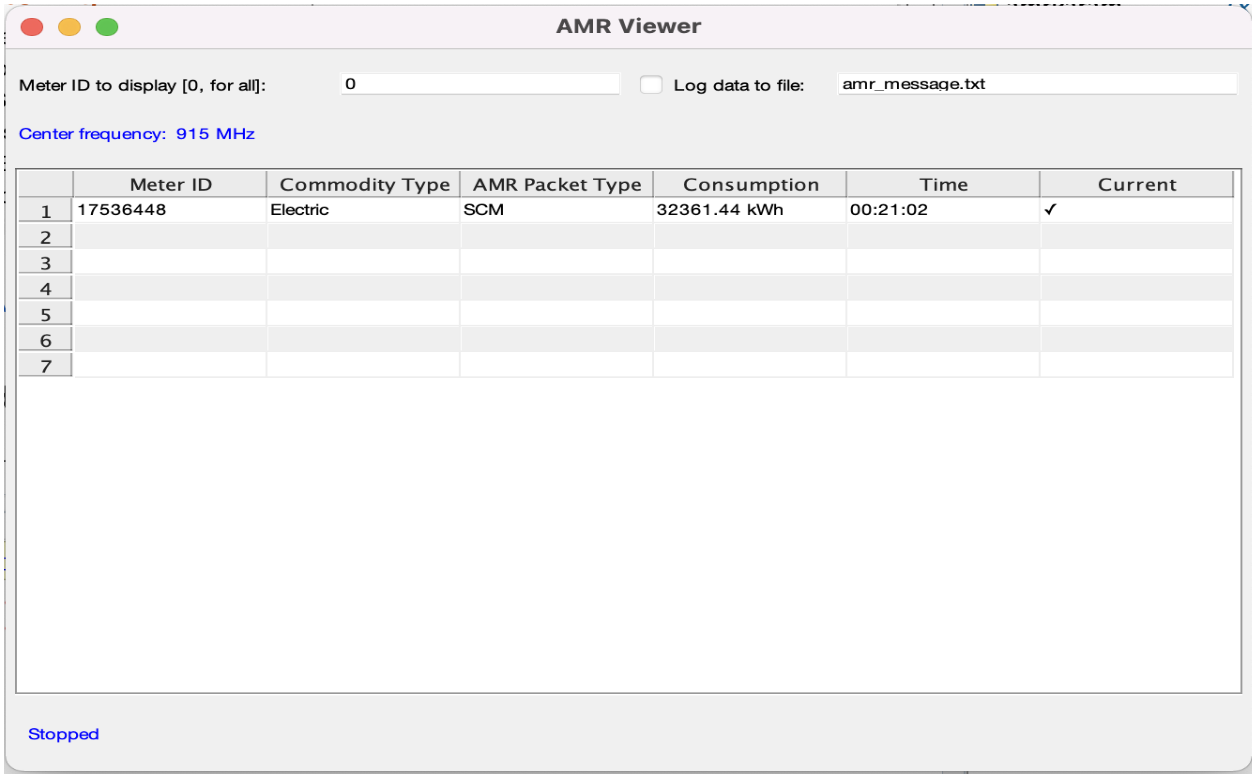
Log data to file: amr_message.txt

Center frequency: 915 MHz

	Meter ID	Commodity Ty...	AMR Packet Ty...	Consumption	Time	Current
1						
2						
3						
4						
5						
6						
7						

Stopped

Fig2. Result using RTL-SDR as a signal resource.


AMR Viewer

Meter ID to display [0, for all]: 0

Log data to file: amr_message.txt

Center frequency: 915 MHz

	Meter ID	Commodity Type	AMR Packet Type	Consumption	Time	Current
1	17536448	Electric	SCM	32361.44 kWh	00:21:02	✓
2						
3						
4						
5						
6						
7						

Stopped

Fig2. Result using a recorded data from Matlab

The first figure is a real-time signal result obtained from using RTL-SDR. Unfortunately, I was not able to get a signal even using the antenna in the Lab. As a result, I was forced to use a recorded file as my source of the signal.

In conclusion, Advanced metering systems can provide benefits for utilities, retail providers, and customers. At first AMR devices just collected meter readings electronically and matched them with accounts. As technology has advanced, additional data could then be captured, stored, and transmitted to the main computer, and often the metering devices could be controlled remotely. This can include event alarms such as tamper, leak detection, low battery, or reverse flow. Many AMR devices can also capture interval data, and log meter events.

Appendix: Code for Automated meter reading:

Note: The default signal source is 'File', which runs the example using the recorded baseband signal file amr_capture_01.bb. To run the example using your RTL or ADALM-PLUTO SDR, change the setting for signalSource when you call the helperAMRInit.m file. Valid options for signalSource are 'File', 'RTL-SDR', and 'ADALM-PLUTO'.

```
signalSource = 'File';
initParam = helperAMRInit(signalSource);

% Calculate AMR system parameters based on the initialized parameters
[amrParam,sigSrc] = helperAMRConfig(initParam);

% Create the data viewer object
viewer = helperAMRViewer('MeterID',initParam.MeterID, ...
    'LogData',initParam.LogData, ...
    'LogFilename',initParam.LogFilename, ...
    'Fc',amrParam.CenterFrequency, ...
    'SignalSourceType',initParam.SignalSourceType);

start(viewer);
radioTime = 0; % Initialize the radio time

% Main Processing Loop
while radioTime < initParam.Duration
    rcvdSignal = sigSrc();
    amrBits = helperAMRRxPHY(rcvdSignal,amrParam);
    amrMessages = helperAMRMessageParser(amrBits,amrParam);
    update(viewer,amrMessages);
    radioTime = radioTime + amrParam.FrameDuration;
end

stop(viewer); % Stop the viewer
release(sigSrc); % Release the signal source
```