Jurnal Penelitian Politeknik Penerbangan Surabaya
Edisi XXXIII, Vol 6, No 3, Bulan September Tahun 2021
p-ISSN: 2615-8671
e-ISSN: 2615-868X

PORTABLE AVIATION NAVIGATION CALIBRATION SYSTEM BASED ON RASPBERRY PI, HACK RF ONE, AND SOFTWARE DEFINED RADIO COMMUNICATION SYSTEM

Ade Irfansyah, Mohamad Rifai, Yuyun Suprapto, Meita Maharani, Bambang Bagus
Politeknik Penerbangan Surabaya

Abstrak

Prosedur kalibrasi dalam sistem navigasi penerbangan yang melibatkan penggunaan pesawat dan *crew* penerbang dapat dipandang sebagai aktivitas yang berisiko dan berbiaya tinggi. Makalah ini memperkenalkan desain sistem komunikasi yang menyediakan solusi berbiaya rendah dan fleksibel, berbasis Raspberry Pi, dan perangkat *Software Defined Radio* HackRF One, yang dilengkapi dengan sensor-sensor yang berhubungan dengan penerbangan. Prinsip-prinsip rancang bangun yang utamanya terdiri dari baik komunikasi yang fleksibel dan kemudahan mengoperasikan, dijelaskan di awal makalah. Metode *frequency shift keying* yang diimplementasikan bersama mekanisme integritas data diterapkan untuk menghasilkan transmisi data sensor yang cepat namun handal. Kecepatan komunikasi data pada lingkungan *single board computer* mencapai 40 kbps. Sistem yang disampaikan juga memiliki keberhasilan pengiriman data sebesar 88%. Selanjutnya, diskusi terkait desain sistem dan pengembangan lebih lanjut juga disajikan dalam makalah ini.

Keywords: Penerbangan, *Software Defined Radio*, *Frequency Shift Keying*, Modulasi Digital, Sistem Sensor.

Abstract

Calibration procedures for the aviation navigation system which involve full size aircraft and its crew may be considered risky and high cost . This paper introduces the state of the art communication system to provide lower cost and flexible solutions based on Raspberry Pi, and SDR device Hack RF One, equipped with proper and aviation-related sensors, which later may be treated as unmanned aerial vehicle payload to replace flying aircraft aviation instruments. Design principles which mainly consist of both flexible communication and operating feasibility are explained in early of this paper. As a preliminary method, simple FSK modulation coupled with data integrity mechanisms are implemented to realize reliable but fast sensor data transmission. Data communication rate under a single board computer environment may achieve 40 kbps. Proposed system achieved a data transmission success rate as high as 88% in reasonable condition. In addition, the discussion of further system development is presented.

Keywords: Aviation, Software Defined Radio, Frequency Shift Keying, Digital Modulation, Sensor System.

INTRODUCTION

Calibration schemes for the aviation navigation system and equipment usually use complex and relatively high cost tools. Full size aircraft are also involved during the calibration procedures. The calibration activities can be seen as simple but may take a long time. The main idea is communication between aircraft instruments and ground equipment which results in several parameters that need to be tuned in the ground side. On the aircraft side, the instrument reads several type sensors and sends them continuously. While observing the parameter viewed in the ground side, the officers are in contact with pilot and aircraft crew frequently in order to tune the equipment based on received parameters. The aircraft may take several times approaching the same airport in order to complete the process. These procedures may be considered as risky, high cost, but very important. However, the advancement of technology on drones, sensors, and communication systems are now available to be composed in order to provide better and lower cost solutions for such conditions.

In recent years, Software Defined Radio [1][2] is considered to be a low cost but powerful solution for communication in the field of research and practical systems. SDR devices are varying and becoming more affordable, while more hardware features and capabilities are always increasing. The presence of free and open source softwares, such as GNU Radio [3] and GNU Radio Companion [4], may open new opportunities in solving problems, especially in designing communication systems. Several works have been done in similar fields of communication, such as researching QAM communication [5], FMCW weather radar [6], and NAV Analyzer [7], all of them utilize SDR and GNU Radio.

The development of single board computers, which are low cost and low energy, such as Raspberry Pi has been helping makers and researchers. Apart from the low price, its computation power is arguably better with the years. The combination of sensors and single board computer has resulted in the enormous solutions in embedded and robotics systems. While most single board computers like Raspberry Pi support Unix based operating system out of the box, which means that it's a perfect match with Software Defined Radio, there is still a lot of opportunity in providing communication systems solutions.

This paper introduces the development of an aviation navigation calibration system, which consists of a sensor system and Raspberry Pi, combined with SDR based device using GNU Radio to establish a real time communication system in order to provide lower cost solution for aviation and navigation calibration procedures. Design principles are explained in the early section to put clear understanding of problems, and requirements, and followed by proposed system design. Results and evaluations are then presented, followed by discussions and conclusions about feasibility of the proposed system.

DESIGN PRINCIPLES

A. Frequency Range

Designed system should have a wide frequency range of communication. The chosen SDR based device as primary communicator is useful compared to *commodity-of-the-self* devices used in related work of NAV analyzer [7]. By using SDR hardware such as Hack RF, up to 6 GHz frequency range is available. Since Hack RF has the ability to be both transmitter and receiver, there is no issue in the compatibility of communicator devices. While designing for current system, a wide frequency range may be considered suitable for further work.

B. Communication Parameters Tunability

Designed system should open for the possibility of tunability, especially for communication parameters. The chosen SDR based communication system is proven for this kind of principles in several related works [6][8][9].

C. Choices of Sensors

Designed system should have sensors suitable for representing navigation and aviation data. The minimum parameters may be used, such as altitude, inertial condition (roll, pitch, yaw), and speed (both vertical and horizontal). Those parameters may be obtained from simple but powerful sensor such as IMU (inertial measurement unit) sensor, gyro, accelerometer, and barometer sensor.

D. Portability

While not strongly mandatory, the portability of the system may be met. The system should provide a terminal or communication port to communicate and obtain both raw and visual data. The host computer, which usually communicates with the system, should need the minimum software installed to be able to access the system.

E. Communication and Data Integrity

Sensors data, which is represented with digital data, must be communicated efficiently and effectively through the air. Radio frequency communication provided by SDR based hardware, may be combined with digital modulation scheme. Considering data loss may occur, designed system must have data integrity scheme coupled with a modulation scheme. Error detection and packetization scheme must be established to make sure no wrong data is received and visualized.

F. *Aviation-Centric* Visualization

Digital data representing the sensor, are better visualized in *aviation-centric* visualization. Primary Flight Display may be brought to display, and provide proper animation responding to received sensor data. Previously mentioned Portability principle, also make sure this kind of visualization can run smoothly in the host computer.

SYSTEM DESIGN

A. Data Producer: Sensor Systems

Proposed system data flow as illustrated in figure 1 begins from IMU and barometric sensor, and ends with PC as operating host. System design is divided into 4 sections: (A) Data Producer, (B) Transmitter, (C) Receiver, and (D) Data Consumer.

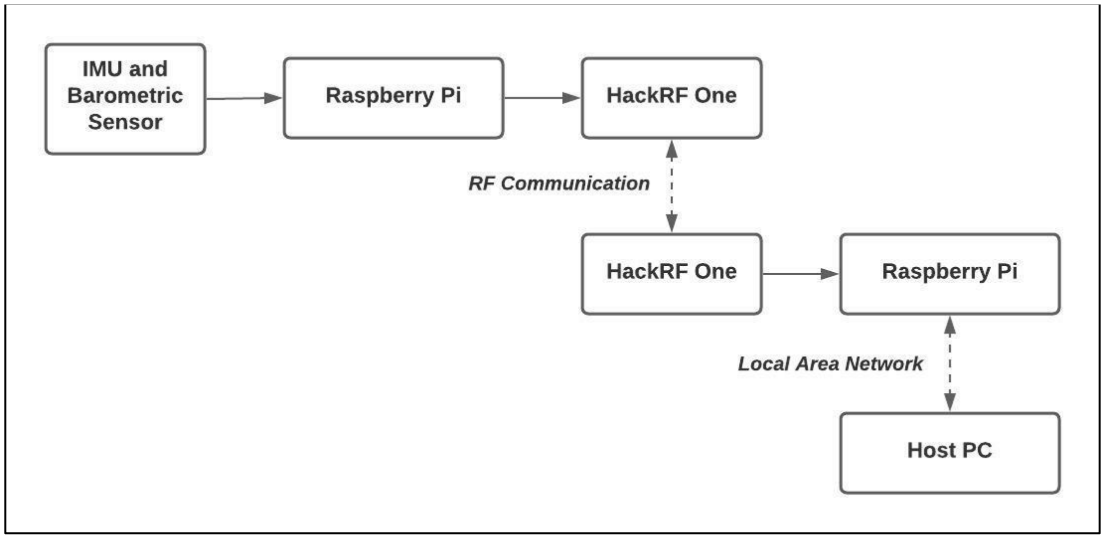


Figure 1. Block diagram of proposed system

Data producer primarily consists of sensors reader and fusion. Data producer utilizes Raspberry Pi 3B+ single board computer which has compute power 1.4 GHz quad core ARM processor. Raspberry Pi is well known for its portability and low energy operation, compared to Intel based commodity-of-the-self computer. Single board computer such as Raspberry Pi is chosen also because of its convenience for interfacing sensor, equipped with various software options on top of its linux based Raspbian operating system.

Table 1. Obtained data specifications

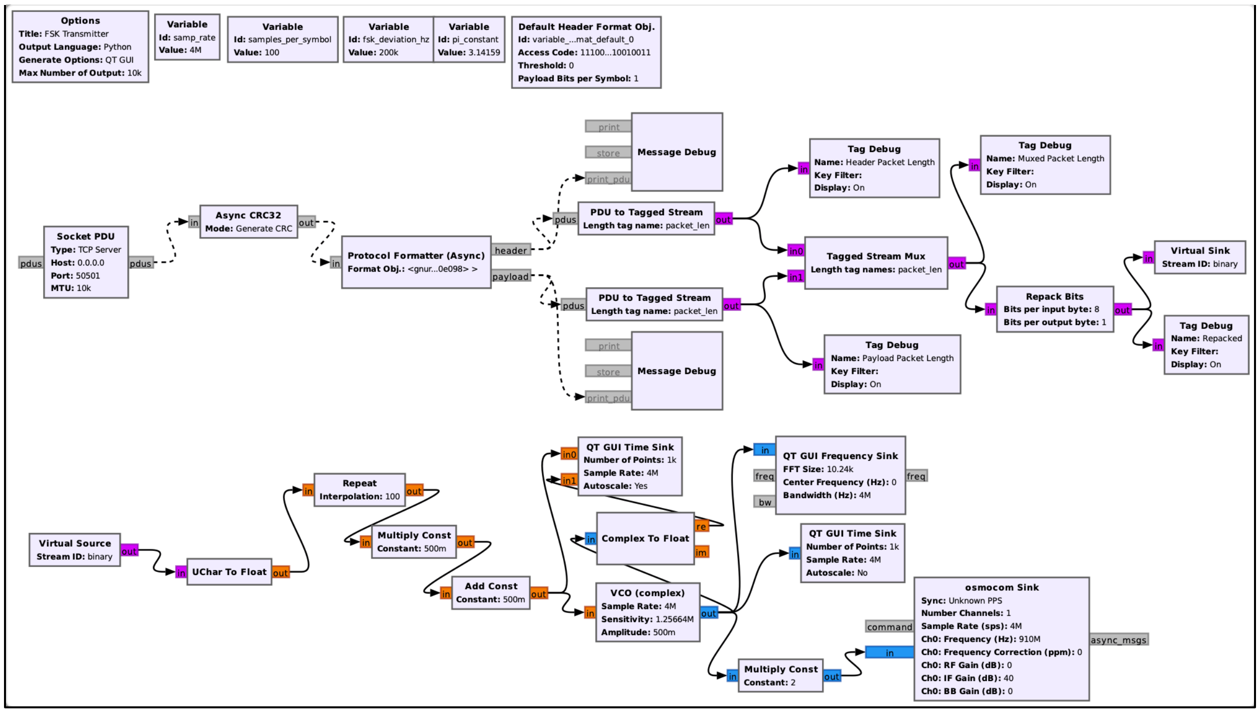
Obtained Data	Source Sensor	Value Unit	Value Range
Roll Rotation	MPU9250 IMU Sensor	Degree	-180.0 until 180.0 deg
Pitch Rotation	MPU9250 IMU Sensor	Degree	-180.0 until 180.0 deg
Yaw Rotation	MPU9250 IMU Sensor	Degree	-180.0 until 180.0 deg
Vertical Speed	MPU9250 IMU Sensor	Knots	0 <
Horizontal Speed	MPU9250 IMU Sensor	Knots	0 <
Altitude	BME280 Barometer Sensor	Feet Above Sea Level	0 <

Two kind of sensor are used to provide up to 6 minimum aviation-related data: (1) MPU 9250 IMU Sensor, and (2) BME 280 Barometer Sensor. Table 1 contains obtained data specifications and their relation with source sensor. While BME280 barometer sensor computes altitude data based on ambience pressure, in straightforward way, the acquisition of five other data come from sensor data fusion techniques. IMU raw sensor data generally consists of accelerometer and gyro data in three axes, X, Y and Z. Sensor data fusion and Kalman Filter operation applied to produce better and representative kind of data such as Roll, Pitch and Yaw. In a similar way, Data Producer produces vertical and horizontal speed data.

B. Transmitter: SDR Based Data Packetization and FSK Transmitter

Transmitter section of the proposed design consists of data packetization and FSK modulator. While an instance of process inside the Raspberry Pi is reading and fusion the raw data, produced final sensor data is forwarded to another process of data packetization and FSK modulator. Transmitter section is designed using GNU Radio Companion software, and compiled into a runnable flowgraph in python executable script as described in [3]. Final flowgraph is shown in figure 2.

At the upstream of flowgraph, Socket PDU block is opening TCP port in Raspberry Pi localhost port 50501. Data producer forwards its result into this port to ensure smooth multi process communication. Multi processing techniques also increase Raspberry Pi core utilizations up to 400% (4 cores x 100%). However, such maximum compute power is seen to be infeasible since it may consume higher power, in unnecessary data producer rate. Finally, multiprocessing is limited to up to half of its maximum compute power, which ranges between 50% until 200%.



Options Title: FSK Transmitter Output Language: Python Generate Options: QT GUI Max Number of Output: 10k
Variable Id: samp_rate Value: 4M
Variable Id: samples_per_symbol Value: 100
Variable Id: fsk_deviation_hz Value: 200k
Variable Id: pi_constant Value: 3.14159
Default Header Format Obj. Id: variable_...mat_default_0 Access Code: 11100...10010011 Threshold: 0 Payload Bits per Symbol: 1
Socket PDU Type: TCP Server Host: 0.0.0.0 Port: 50501 MTU: 10k
Async CRC32 Mode: Generate CRC
Protocol Formatter (Async) Format Obj.: <gnur...0e098> >
header
payload
Message Debug
print
store
print_pdu
PDU to Tagged Stream Length tag name: packet_len
PDU to Tagged Stream Length tag name: packet_len
Message Debug
Tag Debug Name: Header Packet Length Key Filter: Display: On
Tagged Stream Mux Length tag names: packet_len
Tag Debug Name: Payload Packet Length Key Filter: Display: On
Tag Debug Name: Muxed Packet Length Key Filter: Display: On
Repack Bits Bits per input byte: 8 Bits per output byte: 1
Virtual Sink Stream ID: binary
Tag Debug Name: Repacked Key Filter: Display: On
Virtual Source Stream ID: binary
UChar To Float
Repeat Interpolation: 100
Multiply Const Constant: 500m
Add Const Constant: 500m
QT GUI Time Sink Number of Points: 1k Sample Rate: 4M Autoscale: Yes
Complex To Float
VCO (complex) Sample Rate: 4M Sensitivity: 1.25664M Amplitude: 500m
QT GUI Frequency Sink FFT Size: 10.24k Center Frequency (Hz): 0 Bandwidth (Hz): 4M
QT GUI Time Sink Number of Points: 1k Sample Rate: 4M Autoscale: No
Multiply Const Constant: 2
osmocom Sink Sync: Unknown PPS Number Channels: 1 Sample Rate (sps): 4M Ch0: Frequency (Hz): 910M Ch0: Frequency Correction (ppm): 0 Ch0: RF Gain (dB): 0 Ch0: IF Gain (dB): 40 Ch0: BB Gain (dB): 0
command
async_msgs

Figure 2. GNU Radio flowgraph of transmitter section.

Captured data in TCP port 50501 consists of 6 sensor data and 1 index. The series of data is denoted as Payload with N bytes long. Payload which comes into flowgraph is automatically marked with packet length as its header and then continued to the upper series of data packetization blocks, as illustrated in figure 3 stage A and stage B. The flowgraph then computes 32 bit CRC data of payload, and appends it in the front of the data packet as illustrated in figure 3 stage C. Finally, GNU Radio provides 4 bytes unique access code to mark the beginning of data packet.

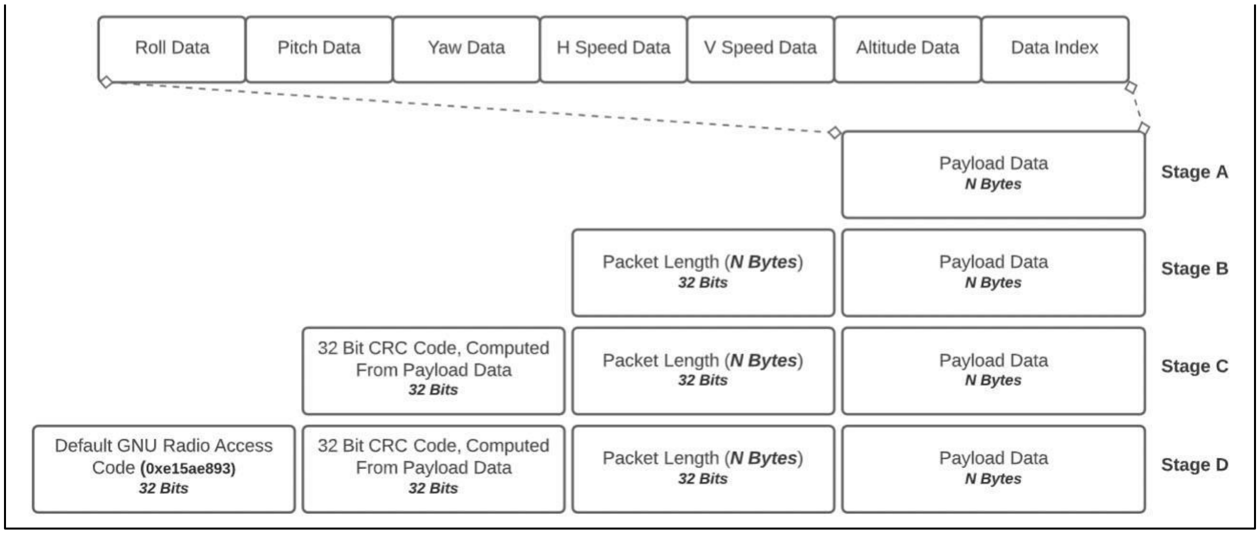

Roll Data
Pitch Data
Yaw Data
H Speed Data
V Speed Data
Altitude Data
Data Index
Payload Data N Bytes
Stage A
Packet Length (N Bytes) 32 Bits
Stage B
32 Bit CRC Code, Computed From Payload Data 32 Bits
Stage C
Default GNU Radio Access Code (0xe15ae893) 32 Bits
Stage D

Figure 3. Data packetization stages.

Data packet came from upper series of blocks is then continued to lower series of blocks which aim to unpack the data as binary, modulate it using Frequency Shift Keying (FSK) method, and ends in *osmocom sink* which connect to *software defined radio* (SDR) hardware, Hack RF one. Voltage Controlled Output is take primary role in modulator blocks, which take input of binary signal 0 and 1, and resulting frequency modulated signal, as shown in figure 4. Sensitivity parameter is configured to has frequency range of 100kHz and 200kHz, which means that binary data of 0 will resulting 100kHz modulated signal, and binary data of 1 will resulting 200kHz modulated signal.

As shown in figure 2, proposed system is using ISM band frequency of 910MHz with sample rate of 4MHz. By using *sample_per_symbol* of 100, transmitter section is designed to send up to 40kbps data. 4MHz of sample rate is chosen to avoid underflow condition, which data producer produce sample data less than SDR hardware capability of consuming the sample. The higher the sample rate, the better and the faster data transmission, ideally, under the limit of computation power of *single board computer* environment.

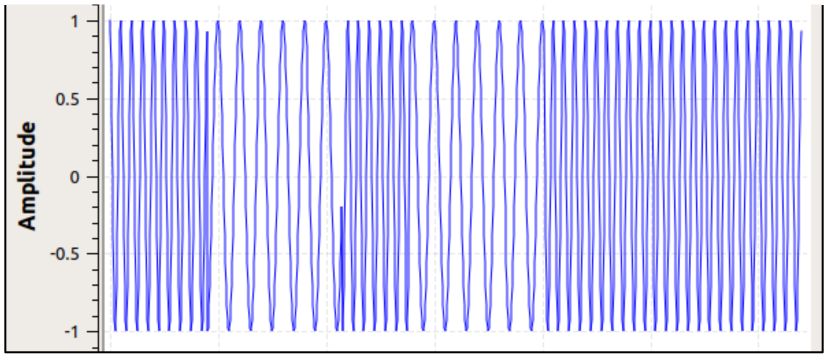
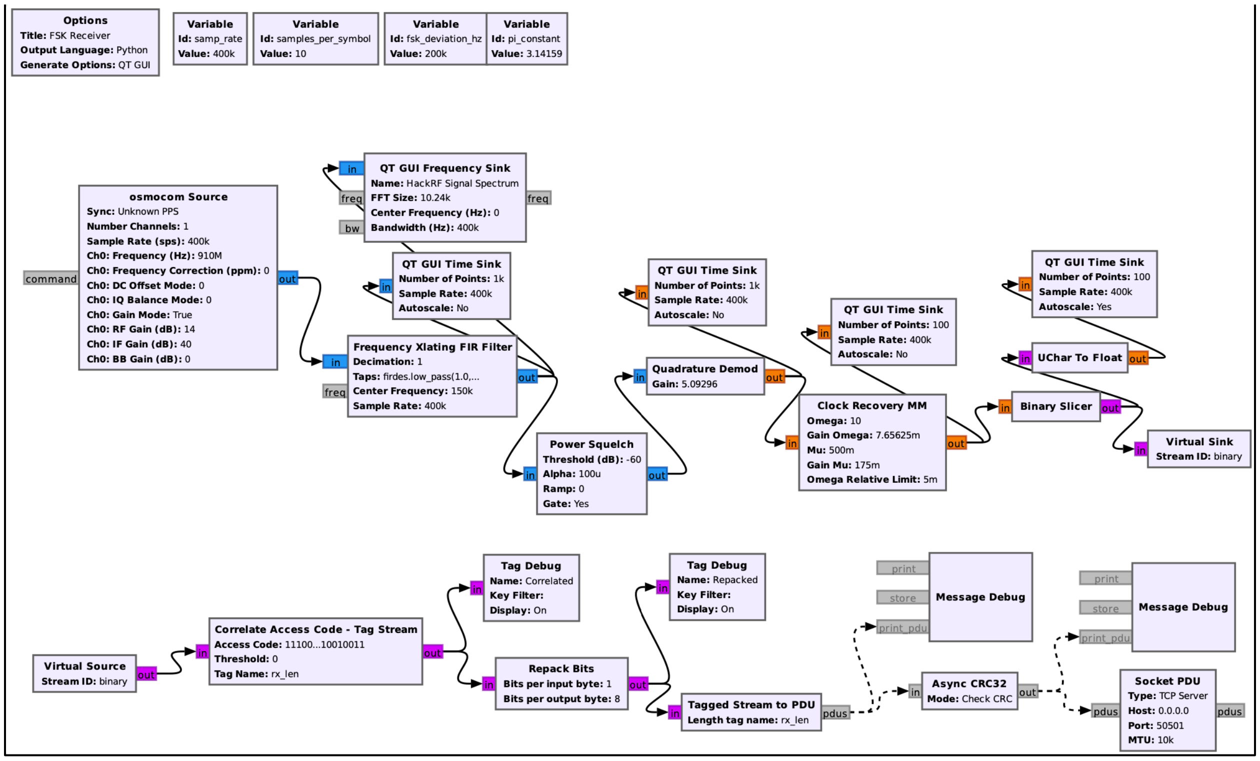


Figure 4. FSK Waveform

C. Receiver: FSK Receiver and SDR Based Data De-packetization

Transmitted data applied into Hack RF One in the transmitter section, is then received by the receiver section. Receiver section tuned to listen at the center frequency of 910MHz, with lower sample rate of 400kHz. Compared to the transmitter section, the receiver is configured to have lower sample rate to reduce the effect of overflow in a *single board computer* environment. Raspberry Pi has limited computation power which impacts its performance to capture real time sample from SDR. This condition ends with the result of overflow, where sample data come from *osmocom source* such as HackRF One is faster than the user process consuming the sample. However, 40kbps of data transmission rate can be synchronized by using lower *samples_per_symbol* variable, utilized by the Clock Recover MM block.



Options
Title: FSK Receiver
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 400k

Variable
Id: samples_per_symbol
Value: 10

Variable
Id: fsk_deviation_hz
Value: 200k

Variable
Id: pi_constant
Value: 3.14159

osmocom Source
Sync: Unknown PPS
Number Channels: 1
Sample Rate (sps): 400k
Ch0: Frequency (Hz): 910M
Ch0: Frequency Correction (ppm): 0
Ch0: DC Offset Mode: 0
Ch0: IQ Balance Mode: 0
Ch0: Gain Mode: True
Ch0: RF Gain (dB): 14
Ch0: IF Gain (dB): 40
Ch0: BB Gain (dB): 0

QT GUI Frequency Sink
Name: HackRF Signal Spectrum
FFT Size: 10.24k
Center Frequency (Hz): 0
Bandwidth (Hz): 400k

QT GUI Time Sink
Number of Points: 1k
Sample Rate: 400k
Autoscale: No

Frequency Xlating FIR Filter
Decimation: 1
Taps: firdes.low_pass(1.0,...
Center Frequency: 150k
Sample Rate: 400k

Power Squelch
Threshold (dB): -60
Alpha: 100u
Ramp: 0
Gate: Yes

QT GUI Time Sink
Number of Points: 1k
Sample Rate: 400k
Autoscale: No

Quadrature Demod
Gain: 5.09296

QT GUI Time Sink
Number of Points: 100
Sample Rate: 400k
Autoscale: No

Clock Recovery MM
Omega: 10
Gain Omega: 7.65625m
Mu: 500m
Gain Mu: 175m
Omega Relative Limit: 5m

QT GUI Time Sink
Number of Points: 100
Sample Rate: 400k
Autoscale: Yes

UChar To Float

Binary Slicer

Virtual Sink
Stream ID: binary

Virtual Source
Stream ID: binary

Correlate Access Code - Tag Stream
Access Code: 11100...10010011
Threshold: 0
Tag Name: rx_len

Tag Debug
Name: Correlated
Key Filter:
Display: On

Repack Bits
Bits per input byte: 1
Bits per output byte: 8

Tag Debug
Name: Repacked
Key Filter:
Display: On

Tagged Stream to PDU
Length tag name: rx_len

Message Debug

Async CRC32
Mode: Check CRC

Message Debug

Socket PDU
Type: TCP Server
Host: 0.0.0.0
Port: 50501
MTU: 10k

Figure 5. GNU Radio flowgraph of receiver section.

Shown in figure 5, Frequency Xlating FIR Filter helps the receiver to limit only to 400kHz bandwidth of signal. Since the sensitivity parameter of VCO block in transmitter is configured to produce frequency range of 100kHz and 200kHz, center frequency is configured to have value of 150kHz. Upper series of receiver blocks aim to demodulate the captured signal and produce binary waveform. Power squelch block reject any signal with power lower than -60dB relative gain. Noise signal should be rejected by this block to reduce undesirable binary data. The series of Quadrature demod, Clock Recovery MM, and Binary Slicer demodulates and shapes the output signal in binary waveform.

Binary signal is first correlated with Access Code to find the mark of the beginning of packet data. The resulted packet data is then repacked, and tagged with found packet length. CRC32 block will separate first 4 byte CRC code with remainder bytes of payload. CRC32 block will drop the payload if the CRC code received and examined is incorrect. Only the correct payload of data will ends in Socket PDU block, in TCP port 50501. Any process may consume the demodulated and correct data to be processed in further methods.

D. Data Consumer: Primary Flight Display Web View Visualization

Data consumer formed as a process running inside Raspberry Pi, consumes receiver-produced data in address localhost:50501, and visualizes the data in real time inside the browser. Web app is chosen as end user visualization method because of its versatility in rendering and animating. Furthermore, accessing the app via web browser is seen as the best way to operate the proposed system, since the host PC doesn't need to install any software. Both portability and *aviation-centric* principle can be met using web app based visualization.

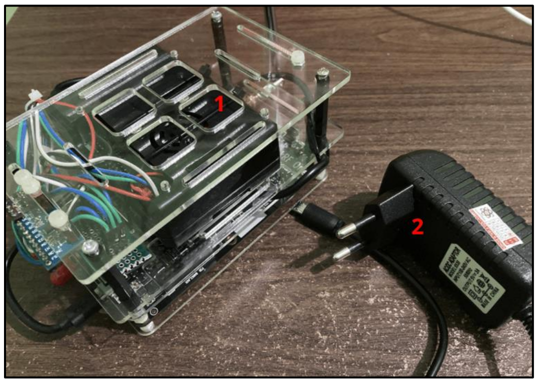


Figure 6. Transmitter device, which consists of (1) sensor systems with Raspberry Pi and Hack RF in a stacked portable package, and (2) power supply adapter.

RESULTS AND EVALUATION

A. PROTOTYPING AND VISUALIZATION

Data producer, sensor system, and FSK modulator run in Raspberry Pi *single board computer* as shown in figure 6. Stacked portable package made by acrylic may be considered as drone or payload of *unmanned aerial vehicle* for later discussions. Raspberry Pi is placed on top of HackRF One, connected to both SDR and sensors. Power supply adapter with rating of 5 volt and 3 amps is used to supply enough power into Raspberry Pi, HackRF and sensors, while doing the experimentation and evaluation.

Receiver section consists of relatively simple components, which are Raspberry Pi, HackRF One, Power Supply Adapter and Cables, as shown in figure 7. Raspberry Pi is then remotely operated by using VNC protocol through either wireless or cable local area network. Both transmitter and receiver configured using static IP Address of 192.168.100.100/24, so that Host PC may connect to them alternately through LAN cables. Though its best to then connect both of them in the same wireless access point to establish better local area network, since Raspberry Pi 3B+ has support for wifi connection.

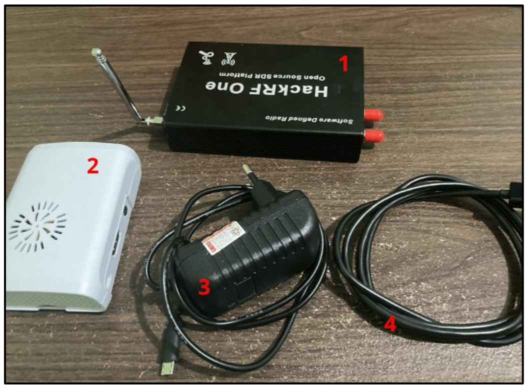


Figure 7. Receiver device, which consists of (1) Hack RF, (2) Raspberry Pi, (3) power supply adapter, and (4) USB communication cable.

Raspberry Pi on the receiver side is configured to open web service port 7800 which is accessible through a web browser. By visiting ***http://<receiver ip address>:7800***, Primary Flight Display visualization can be displayed automatically. The values and animation will react to the receiving mechanism in the data consumer side. Figure 8 shows how roll rotation of transmitter device ends with result of roll animation in Primary Flight Display.

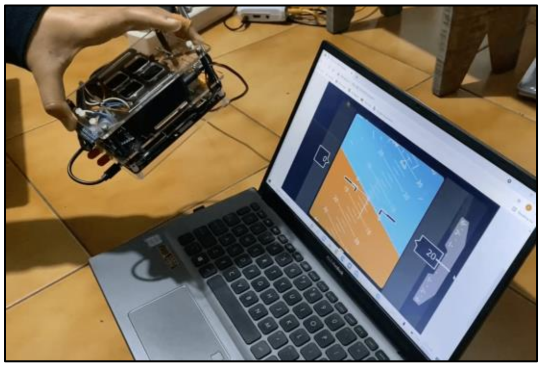


Figure 8. Primary Flight Display in action, receiving and visualizing real time data from the transmitter.

B. SENSORS EVALUATION

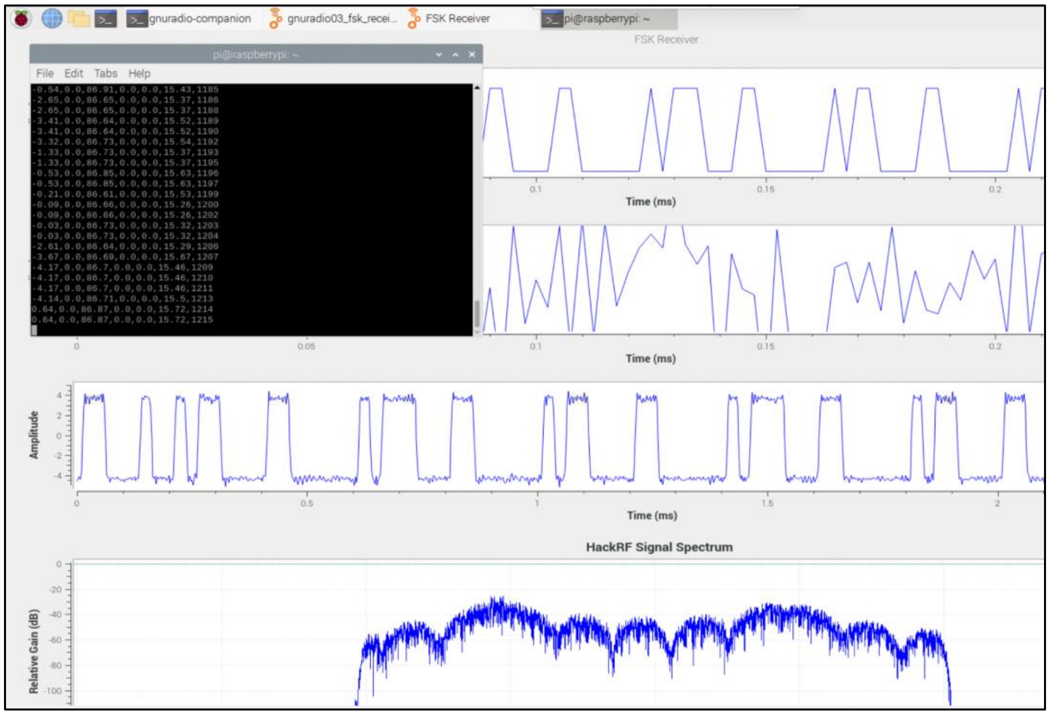
Conducted measurement result for 6 *aviation-related* data is shown in figure table 2. Rotation measurement produces quite good results with relatively lower average error numbers. The remaining three data yield relatively higher average error. This fact is related to the availability of sensor fusion and notable kalman filters for rotation data, but not for remaining other parameters. However the result also provides good establishment at this preliminary development, since the chosen sensors are operating properly. Both varying kind of sensors and developing data acquisition and filter methods are considered as further work which may provide lower average error and broader *aviation-related* parameters.

Table 2. Obtained data average error.

Obtained Data	**Value Unit**	**Average Error**
Roll Rotation	Degree	0.359 degree
Pitch Rotation	Degree	0.04975 degree
Yaw Rotation	Degree	0.19075 degree
Vertical Speed	Knots	3.117 knots
Horizontal Speed	Knots	3.257 knots
Altitude	Feet Above Sea Level	0.358 feet

C. COMMUNICATION RESULT

FSK based data transmission is done successfully, as shown in figure 9, Raspberry Pi terminal listening to local port of 50501 prints captured sensor data, delimited with "," (comma) character. GNU Radio real time waveform displayed behind the terminal, from the top chart to bottom, shows (1) binary representation of signal, (2) demodulated signal, (3) symbol recovered signal, and (4) received signal spectrum.



```
-0.54,0.0,86.91,0.0,0.0,15.43,1185
-2.65,0.0,86.65,0.0,0.0,15.37,1186
-2.65,0.0,86.65,0.0,0.0,15.37,1188
-3.41,0.0,86.64,0.0,0.0,15.52,1189
-3.41,0.0,86.64,0.0,0.0,15.52,1190
-3.32,0.0,86.73,0.0,0.0,15.54,1192
-1.33,0.0,86.73,0.0,0.0,15.37,1193
-1.33,0.0,86.73,0.0,0.0,15.37,1195
-0.53,0.0,86.85,0.0,0.0,15.63,1196
-0.53,0.0,86.85,0.0,0.0,15.63,1197
-0.21,0.0,86.61,0.0,0.0,15.53,1199
-0.09,0.0,86.66,0.0,0.0,15.26,1200
-0.09,0.0,86.66,0.0,0.0,15.26,1202
-0.03,0.0,86.73,0.0,0.0,15.32,1203
-0.03,0.0,86.73,0.0,0.0,15.32,1204
-2.61,0.0,86.64,0.0,0.0,15.29,1206
-3.67,0.0,86.69,0.0,0.0,15.67,1207
-4.17,0.0,86.7,0.0,0.0,15.46,1209
-4.17,0.0,86.7,0.0,0.0,15.46,1210
-4.17,0.0,86.7,0.0,0.0,15.46,1211
-4.14,0.0,86.71,0.0,0.0,15.5,1213
0.64,0.0,86.87,0.0,0.0,15.72,1214
0.64,0.0,86.87,0.0,0.0,15.72,1215
```

Figure 9. Carrier signal observation result and received raw sensor data in the receiver section.

Spectrum shown in figure 9, contain two highest peaks as a pair of binary 0 frequency and binary 1 frequency, which are 100kHz and 200kHz. Centered frequency of 150kHz with bandwidth of 400kHz captures the data signal successfully, then transforms them into binary representation. The spectrum also contain other undesirable peaks which come from the implementation of BFSK modulation. Several types of digital modulation such as Gaussian FSK, n-PSK, and OFDM are considered as further work and theoretically may yield better results in data transmission.

Each line printed in Raspberry Pi terminal contains a data index, placed at the end. Since data index is incremental data, data transmission success rate is calculated by counting data index compared to maximum number of data index every second. Communication system has average packet loss of 12%, which means data transmission success rate is as high as 88%.

CONCLUSIONS AND FUTURE WORK

State of the art implementation is conducted and the transmitter and receiver system are operating properly. Data transmission speed of 40kbps is sufficient to provide real time sense of data communication between data producer and data consumer. End visualization of Primary Flight Display emphasizes the output which has to be aviation-related. Real time animation and value changes are successfully established, displayed in Host PC. Design principles for the proposed system are considered fulfilled.

While the proposed system is considered ready to be the payload for either drone or UAV, several works still have to be done in the near future. Implementing other digital modulations, applying sensor reading and filter, and developing both mechanical and electronic, will support the system for proper payload. Future works also open substantial opportunity for research in related fields.

REFERENCES

J. Bard, and V. Kovarik, “Software Defined Radio: The Software Communications Architecture,” Wiley Series in Software Radio, 2007.

J. Reed, “Software Radio: A Modern Approach to Radio Engineering,” Prentice Hall, 2005.

E. Blossom, “GNU radio: tools for exploring the radio frequency spectrum,” Linux Journal, 2004.

J. D. Edgcombe, B. E. Dunne, “Implementation of Analog and Digital Communications Transceivers on SDR Platforms using GNU Radio Companion,” Proceedings of American Society for Engineering Education 126th Annual Conference & Exposition, 2019.

N. Pambudiyanto, B. B. Harianto, and A. S. Prabowo, “DESAIN KOMUNIKASI QAM (QUADRATURE AMPLITUDE MODULATION) MENGGUNAKAN GNU RADIO,” Jurnal Penelitian Politeknik Penerbangan Surabaya, vol. 5, no. 4, Dec, 2020.

A. Prabaswara, A. Munir, and A. B. Suksmono, “GNU Radio based software-defined FMCW radar for weather surveillance application,” Proceedings of The 6th International Conference on Telecommunication Systems, Services, and Applications, 2011.

M. Rifa’i, R. D. Puspita, O. K. Sektianggi, and R. A. Wulandari, “RANCANG BANGUN ULTRA HIGH FREQUENCY (UHF) TRANSCEIVER DATA NAV ANALYZER BERBASIS ESP32 PADA RPAS (REMOTELY PILOTED AIRCRAFT SYSTEM),” Jurnal Penelitian Politeknik Penerbangan Surabaya, vol. 5, no. 3, Dec, 2020.

A. E. Elsaghier, N. S. Tezel, and S. M. Altiraiki, “Frequency Shift Keying Scheme to Implement SDR using Hackrf one,” International Journal of Electronics Engineering Research, vol. 9, no. 8, 2017, pp. 1147-1157.

S. Hemalatha, Sk. Imran, T. Nirupa, Y. Syam Kishore, B. Chandra Mohan, “Implementation of FSK Transceiver using Software Defined Radio (SDR),” International Research Journal of Engineering and Technology (IRJET), vol. 5, no. 3, 2018, pp. 583-586.