Project: Aerial Drone Notification System (SkyWarden) developed in the Spring 2018 semester at the University of Nevada, Reno

Team members: Rony Calderon, Bryan Kline, Jia Li, and Robert Watkins

Advisors: Dr. Kostas Alexis & Frank Mascarich

Subsystem: On-board Unit

Description: Documentation for the on-board subsystem unit that includes design concept, feature implementation, wiring and connection setups for each sensor module.

## *On-board Subsystem Overview*

The Aerial Drone Notification System (ADNS) on-board subsystem is comprised of a Teensy 3.2 ARM microcontroller, 14 Time of Flight (ToF) sensors, an INA219 current/voltage sensor, and a nRF24L01 2.4 GHz wireless transceiver unit.

The Aerial Drone Notification System is composed of two separate systems that communicate via radio frequency (RF) wireless using the NRF24L01 transceiver modules. The two systems considered are the on-board unit mounted to an aerial drone and a ground base unit that is used to capture the data from the on-board system and process the data accordingly. The system is designed to accommodate up to 20+ VL53L0X ToF sensors using independent addressing through the I2C bus. The ADNS system currently uses 14 ToF sensors and an INA219 voltage/current sensor to grab voltage levels from the aerial drone's battery pack.

The sensors are sequentially polled and transmitted wirelessly from the on-board system to the ground base system in a stream of data that is, unfortunately, unreliable (think UDP). To combat the unreliable datagrams, the ADNS team chose to send each data stream independently. This is done to reduce the amount of bad or corrupted datagrams as well as increase the throughput of the wireless system. Please read the documentation at the website listed below for more information:

http://www.airspayce.com/mikem/arduino/RadioHead/classRH_NRF24.html

Features for the ADNS system include independent I2C addressing and high speed polling rates of the ToF sensors. Currently and collectively, the ToF sensors poll at about 56 Hz. The wireless data rate has been set to the lowest setting possible at 250 kbps to extend the possible ranges of the NRF24L01 transceivers and due to the bottleneck of the system which are the ToF sensors. Additionally, the on-board system has a software reset feature that will attempt to restart the system if a sensor has timed out. This presents a good robust and recovery method but also introduces the possibility of a single sensor triggering a constant reboot of the on-board system. The estimated reboot time is around 500ms (half a second).

Another feature implemented is the auto restart of the system in the event of system initialization failure. The ADNS system begins by trying to poll a set amount of readings for the system upon boot and will trigger a red LED alarm if even one sensor fails to initialize properly. The set amount of readings or polls can be set by the user and default is 250 readings. Additionally, if the

ADNS system successfully initializes, a green LED is turned on constant state to indicate successful initialization. Moreover, if a sensor is not seated properly, the user will see the triggered alarm and can re-seat the faulty sensor to re-initialize the system.

**WARNING:** Please use the system files included with the ADNS including its source code as the source code for several libraries has been changed to implement some of the features.

**WARNING:** Different breakout boards using the VL53L0X ToF sensor may be used interchangeably but the breakout boards must have the XSHUT pin. The XSHUT pin is used for independent addressing when interacting with the I2C bus.

### *ADNS On-board Transmitter System Notes & Pin Configurations*

Since the on-board system for the Aerial Drone Notification System (ADNS) utilizes a Teensy 3.2 microcontroller, we will define the setup of the on-board system specific to the Teensy 3.2 MCU. Other MCU's work as well but the Teensy 3.2 gave the ADNS team enough output digital pins to connect the large amount of sensors placed on the on-board system (20+ sensors will be used). Thus, the pin mapping may not be identical to other boards so please use this precaution to ensure proper connection to the various sensors utilized.

NRF24L01 Transceiver module NOTE: The NRF24L01 transceiver unit has a in-chip voltage regulator and the Vcc pin is 5 volt tolerant. However, 3.3 volts works on some models and does not work in other models. Therefore, if the 3.3V pin is utilized and wireless transmission is not available, please consider switching to the 5V pin. The team encountered this issue with on transceiver module and lessons were learned.

Additionally, the NRF24L01 module works with the 2.4 GHz wireless ISM band and Wi-Fi conflicts & contention are possible. Therefore, the ADNS development team decided to have the NRF24L01 transceiver modules work above the Wi-Fi known operating ranges of 2.401 GHz to 2.495 GHz in accordance to 802.11 legacy ISM protocols and referencing:

http://www.radio-electronics.com/info/wireless/wi-fi/80211-channels-number-frequencies-bandwidth.php

https://www.intel.com/content/www/us/en/support/articles/000005725/network-and-i-o/wireless-networking.html

Due to the possible 2.4 GHz conflicts and contention, the recommended range of channels for the NRF24L01 transceiver is 100 to 125. Any values below channel recommendations can lead to wireless communication issues and any channel values above 125 will default to channel 1.

The wireless RF transmission rates are defined in the macros section with the possible rates of 250 kbps, 1 Mbps, and 2 Mbps. To simplify changing of transmission data rate, an additional macro (TRANSMIT_DATA_RATE) is defined and stores the numerical constant value needed

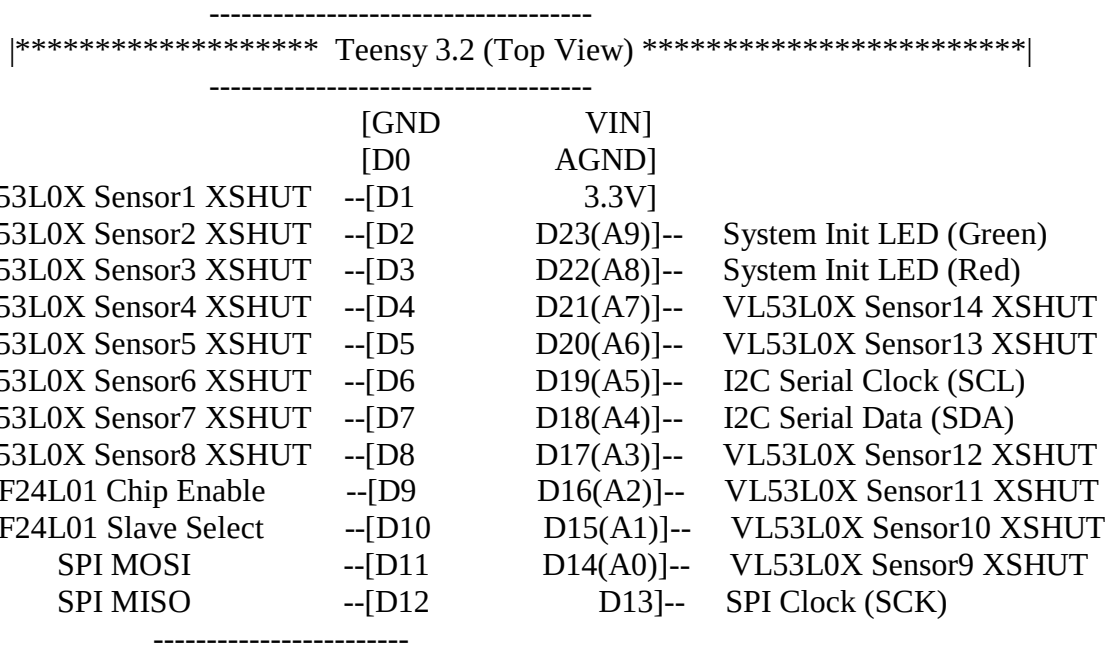
by the NH_NRF24.h library. Please only consider these valid macro values when setting transmission rates.

**WARNING:** The channel and RF wireless transmission rates must match on both the transceiver sides. Namely, the on-board transceiver unit and the base unit must have these parameters matching.

### *ADNS Teensy 3.2 Pin Mapping & Configuration*

**NOTE:** The XSHUT pins are interchangeable for each VL53L0X sensor used but must be configured properly in the *setup()* function to accommodate the new changes. The SPI pins are not interchangeable (MOSI, MISO, & SCK) and must be connected in the specified manner for the NRF24L01 transceiver to work properly. However, the chip-enable (CE) and slave select (SS) pins can be changed to any other available digital IO pin but must be redefined in the macros definitions.

D = Digital pin
(A)= Analog pin



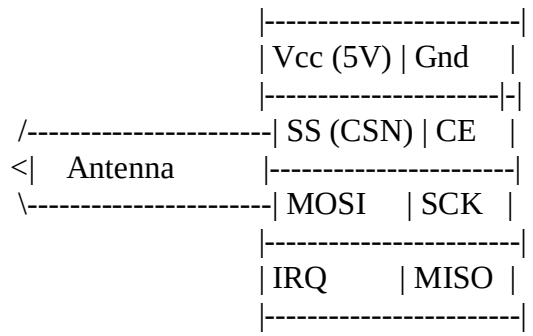

### *NRF24L01 Pin Mapping & Configuration*

**NOTE:** The chip-enable (CE) and slave select (SS) pins can be any digital IO pin but the SPI pins must be mapped to the board such as the following, where A(x) is the alternate pin (if available) and x is the pin number:

**NOTE:** If A(x) is listed, then any digital IO pin may be used.

| SPI Function | ATmega Pin # | Teensy 3.2 Pin # | Nano Pin # |
|---|---|---|---|
| Chip Enable (CE) | 8 A(x) | 9 A(x) | 8 A(x) |
| MISO | 50 | 12 A(8) | 12 |
| M0SI | 51 | 11 A(7) | 11 |
| SPI Clock (SCK) | 52 | 13 A(14) | 13 |
| Slave Select (SS) | 53 | 10 | 10 |

---------- NRF24L01 Transceiver unit with antenna ----------




IRQ pin is used to trigger an interrupt in an MCU but not needed as a steady stream of sensor data is desired.

***INA219 DC Current & High-Side Voltage Sensor Notes and Configuration***

The ADNS development team chose to use the INA219 Current Sensor breakout board to measure the operating current and voltages of the aerial drone which the on-board ADNS system interacts with. The module can be used to measure voltages up to 26V DC and currents of up to 3.2A. The battery's positive and negative terminals are connected via the Vin- and Vin+ pins or Vin- and Vin+ screw ports as depicted below. If current measurements exceed 3.2A, the datasheet recommends reducing the 0.1 ohm current sense resistor and replace it with a 0.01 ohm to reach current measurements of up 32A with a resolution of 8mA. The current sense resistor is also depicted below.

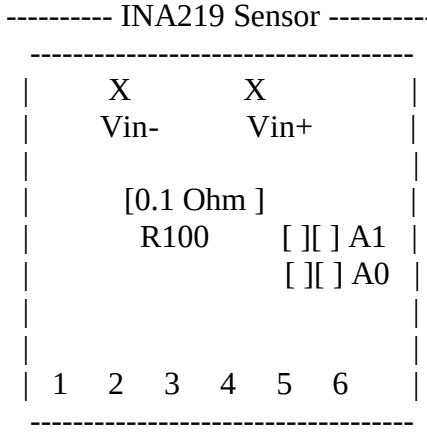
**I2C Address NOTE**: The INA219 breakout broad used for the on-board ADNS system uses the default I2C address associated with the module. Namely, 0x40 is used as the default address but can be changed in the event of I2C address confliction or if multiple INA219 modules are in use. To change the address of any INA219 module, you have to simply solder together the two contact points on A0, A1, or both. Below, are the following combinations allowed:

| I2C Address | A0 Soldered? | A1 Soldered? |
|---|---|---|

| | | | |
|---|---|---|---|
| 0x40 | No | No | |
| 0x41 | Yes | No | |
| 0x44 | No | Yes | |
| 0x45 | Yes | Yes | <---- Both A0 & A1 are soldered and bridged together. |

***INA219 Current & High-Side Voltage Sensor Pin Configuration***

X = Screw terminals
1 = Vcc (3V - 5V)
2 = Gnd
3 = I2C Serial Data (SDA)
4 = I2C Serial Clock (SCL)
5 = Vin- (Alternate from screw terminal Vin-)
6 = Vin+ (Alternate from screw terminal Vin+)




The current sense resistor can be replaced with a lower value to enable greater currents and voltages to be read. The battery's positive lead will be either connected to pin 6 or to the Vin+ screw terminal. The battery's negative lead is connected to ground.

***On-board Program Libraries***

The following program libraries are necessary for a successful initialization of the on-board subsystem and for wireless transmission of the pertinent data:

- ***Wire.h***
  - This library initializes and manages the $I^2C$ bus for the VL53L0X ToF sensors and the INA219 current/voltage battery sensor.

- ***VL53L0X.h***

- Library used for setting and managing the registers of the VL53L0X ToF sensors.

- ***RH_NRF24.h***
  - Library used for setting and managing the registers of the nRF24L01 transceiver. The RH_NRF24.h header file relies on the SPI.h library for SPI communication.

- ***SPI.h***
  - Library used in conjunction with the RH_NRF24.h library to allow for SPI communication.

- ***String.h***
  - Library used to create and manage strings.

- ***Adafruit_INA219.h***
  - Library used to setup and manage the INA219 current/voltage sensor registers.

***On-board Methods***

***File: Wire.h***
***Class:*** Wire
***Methods:***

| | |
|---|---|
| Method Name: | begin() |
| Description: | Initializes and starts the $I^2C$ bus |
| Parameter: | none |
| Return: | none |

***File: RH_NRF24.h***
***Class:*** RH_NRF24
***Methods:***

| | |
|---|---|
| Method Name: | RH_NRF24::RH_NRF24 |
| Description: | Constructor for the nRF24L01 transceiver module |
| Parameters: | CE – The chip-enable pin |
| | SS – The slave select pin |
| Return: | Boolean – True if successful. Otherwise, false |

| | |
|---|---|
| Method Name: | init() |
| Description: | Initialized nRF24L01 transceiver registers |
| Parameter: | none |
| Return: | Boolean – True if successful. Otherwise, false |

| | |
|---|---|
| Method Name: | setChannel() |
| Description: | Sets the wireless channel for the transceiver |
| Parameter: | CH – The channel from 1 – 125 to be set |
| Return: | Boolean – True if successful. Otherwise, false |

| | |
|---|---|
| Method Name: | setRF() |
| Description: | Sets the wireless transmission properties such as data rate |
| Parameter: | TransmitDataRate – The data rate in kbps or Mbps<br>TransmitPower – The transmit power in dB |
| Return: | Boolean – True is successful. Otherwise, false |

| | |
|---|---|
| Method Name: | send() |
| Description: | Sends the character array to the ground unit transceiver |
| Parameter: | msg* – The pointer message to send<br>msgSize – The size if the message |
| Return: | none |

***File: VL53L0X.h***
***Class:*** VL53L0X
***Methods:***

| | |
|---|---|
| Method Name: | VL53L0X::VL53L0X |
| Description: | Constructor for the ToF sensor |
| Parameter: | none |
| Return: | none |

| | |
|---|---|
| Method Name: | init() |
| Description: | Initializes ToF sensor to default settings |
| Parameter: | true – Boolean value to state we are using this ToF sensor |
| Return: | none |

| | |
|---|---|
| Method Name: | setAddress() |
| Description: | Sets the $I^2C$ address on the selected ToF sensor |
| Parameter: | addr – The address (unsigned integer) of the ToF sensor |
| Return: | Boolean – True if successful. Otherwise, false |

| | |
|---|---|
| Method Name: | setTimeout() |
| Description: | Sets the timeout limit in milliseconds for the ToF sensor |
| Parameter: | limit – The timeout limit in milliseconds |
| Return: | none |

| | |
|---|---|
| Method Name: | setVcselPulsePeriod() |

| | |
|---|---|
| Description: | Sets the laser pulse period in PCLKs for the pre and post ranges |
| Parameter: | The limit range in PCLKs |
| Return: | none |

| | |
|---|---|
| Method Name: | setMeasurementTimingBudget() |
| Description: | Sets the polling timing budget for each ToF sensor |
| Parameter: | The timing budget in microseconds |
| Return: | none |

| | |
|---|---|
| Method Name: | startContinuous() |
| Description: | Starts the ToF sensor in continuous polling mode |
| Parameter: | start_in_ms – The time to start the next polling sequence |
| Return: | none |

| | |
|---|---|
| Method Name: | readRangeContinuous() |
| Description: | Reads the contents of the continuous reading register |
| Parameter: | none |
| Return: | The distance value from sensor to object in millimeters |

***File: Adafruit_INA219.H***
***Class:*** INA219
***Methods:***

| | |
|---|---|
| Method Name: | start() |
| Description: | Initializes and starts the INA219 module with default settings |
| Parameter: | none |
| Return: | none |

| | |
|---|---|
| Method Name: | getBusVoltage() |
| Description: | Gathers the bus voltage value on the register |
| Parameter: | none |
| Return: | Returns the float voltage value |

***On-board Program Structure***

The on-board subsystem code design uses many macros to enable users to easily change the values of the subsystem. This includes:

- Modifying the subsystem initialization time, the amount of ToF sensors mounted to the subsystem, the ToF sensor timeout values, and the output baud rate.
- Modification of the ToF operation properties such as the timing budget can be increased or decreased for faster or slower polling,

- Modification of the nRF24L01 wireless properties such as changing the wireless channel used for transmission of data and setting the wireless transmission data rate.

The program begins by initializing the nRF24L01 transceiver SPI pins and the VL53L0X ToF sensor array used to contain the sensor objects for easy initialization and changing the amount of sensors is hassle-free. Moreover, each ToF sensor is assigned a character identifier to help distinguish the values polled and sent to the ground unit for efficient parsing. Next, the battery sensor unit is set to default mode and the $I^2C$ address remains unchanged only for the INA219 sensor.

In the *setup()* function, the transceiver properties are first set to ensure that wireless transmission is ready to go before the sensors begin polling and in the event of an error during initialization. Once the transceiver is successfully initialized, the pins for each ToF sensor is set to *INPUT* mode to enable $I^2C$ address changes for each individual sensor as two identical addresses will cause the $I^2C$ bus to not poll properly. Afterwards, the sensors are each individually initialized and a new $I^2C$ address is assigned in a sequential manner using the *setAddress()* function.

Once the addresses are set, the ToF sensor properties can then be set individually for each sensor. The first property set is the signal rate limit for each sensor using the *setSignalRateLimit()* function. The signal rate is used to determine the light pulse of the LiDAR emitting laser during each distance calculation. Next, the period selection pre-range and post-ranges are set using the *setVcselPulsePeriod()* function. The pre-post limit can be as high as 18 pulse clocks (PCLKs) and not lower than the post-range limit. The lowest post-range limit is 8 but setting the pre-range to 18 and the post-range limit to 14 allows for longer range detection of the ToF sensors. However, higher limit ranges decrease accuracy at longer distances.

Lastly, the ToF sensor timing budget is set using the setTimingBudget() function. The defaults is 33-ms and lowest is slightly under 18-ms. The lower limit allows for faster polling of the ToF sensors but reduces accuracy and the higher limit polls slower but at a high accuracy rate.

Once the sensors are set, a boolean flag, system_initialized_properly) is set and sequential polling of the ToF sensor and INA219 bus voltage is ready to begin.

***Main Program Loop***

The main program is rather simple as it is designed to poll each sensor in a sequential manner and also transmits the data to the ground unit in the same fashion. The loop stores the value gathered from the *readRangeContinuousMillimeters()* function with each sensor in the ToF array. The program initially checks that the value returned is at an acceptable limit since a timeout on any of the sensors returns the timeout value of 65535. If a timeout occurs, the sensor is considered nonresponsive and program attempts to recover by checking for additional timeouts. If the timeout is still evident, the $I^2C$ bus must be restarted and the system is re-initialized using a macro command software reset. If the value is determined to be valid, the program casts the value into a string and concatenates the unique sensor identifier before being sent to the ground unit.

During the packaging of the ToF sensor value, the new string value is converted into a *uint8_t* character array and copied byte by byte using the *toCharArray()* function to ensure the entire message is packaged in one packet. Lastly, the entire message is cast as a pointer and sent through the *send()* function. Once the ToF sensors are polled and transmitted to the ground unit, the main loop finally polls the bus voltage of the attached battery using the *getBusVoltage()* function and performs similar casting to send the value to the ground unit. Before exiting the polling loop, the *SYSTEM_INIT_POLLS* value is incremented to reflect a successful poll. If the value exceeds a user-defined limit for a successful system initialization, a small green LED is activated to show the user visually that the system initialization stage was completed successfully. If the system timed out and is not able to recover in time, the system will instead flash a red LED indicating that the system failed to initialize properly. If system initialization failed, the output reading to the console display the readings gathered by each sensor to the user for easy debugging purposes. For example, if ToF sensor #4 timed out and caused the failure, its timeout value of 65535 will be displayed in the sensor value vector on the output console screen.

## On-board subsystem Schematic