Critical Design Review

Firmware Team

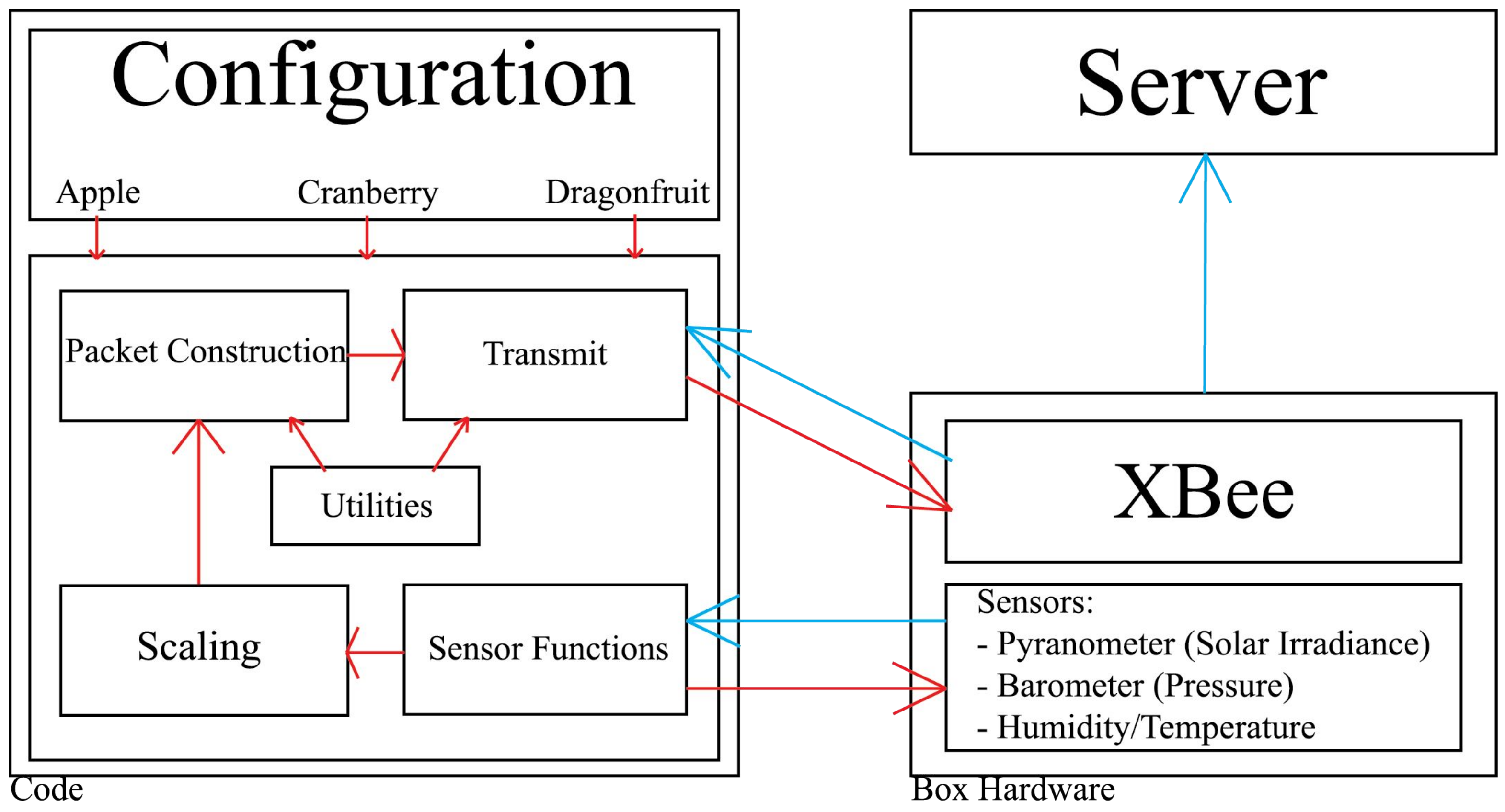
Scott Nakashima
Ryan Walser

Overview

- ▷ Goals
- ▷ Overall Design
- ▷ Implementation
- ▷ Progress
- ▷ Remaining tasks

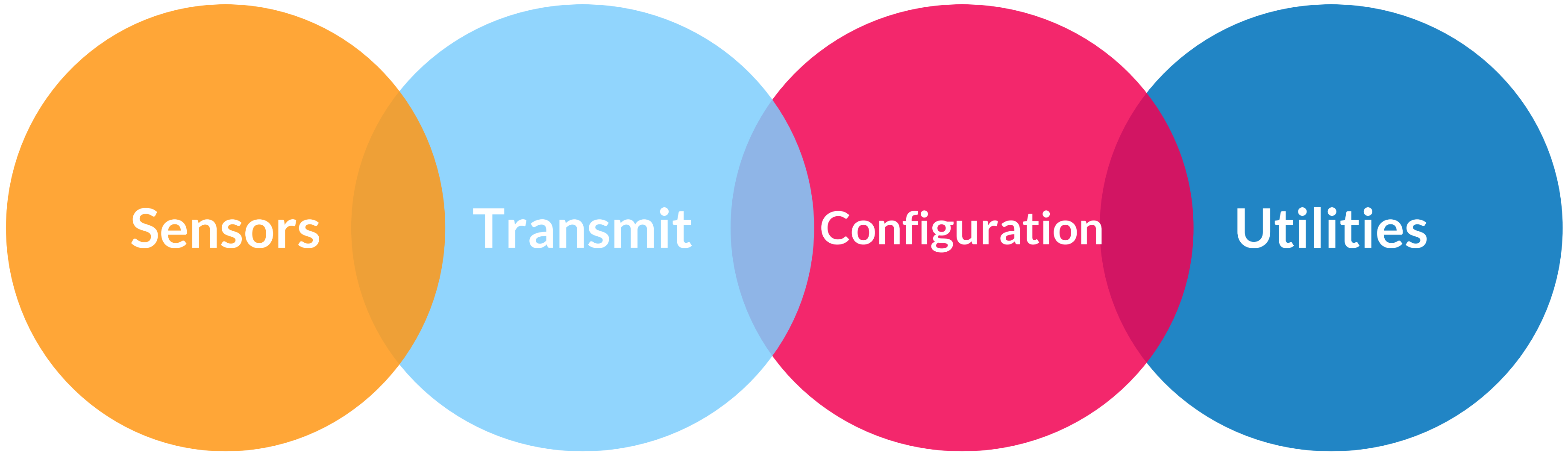
Goals (Fall 2015)

- ▷ Working code
    - ○ Verifiable as working with Apple
- ▷ Easily implementable configurations for other generations
- ▷ Unit Tests



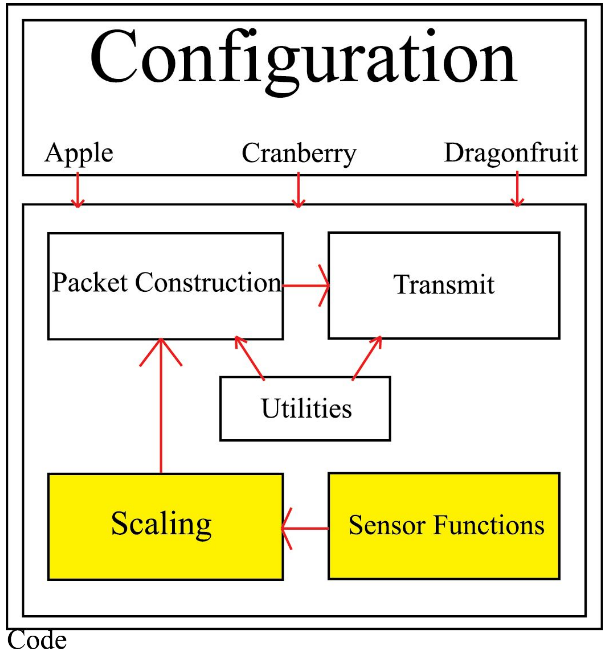
Overall Design

Implementation

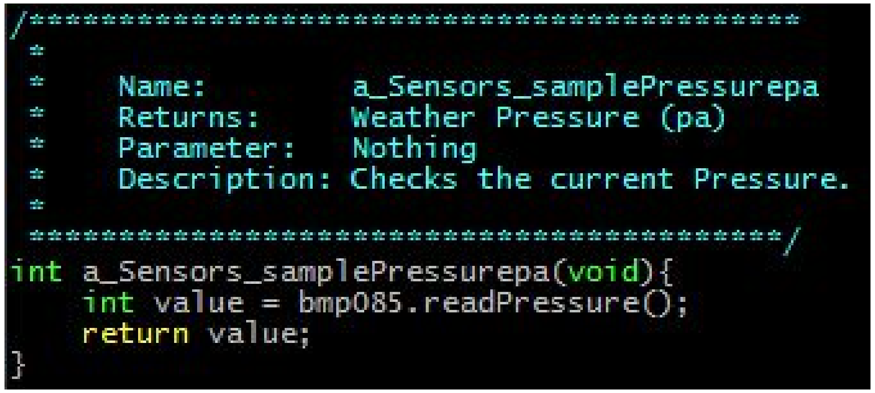


Sensors

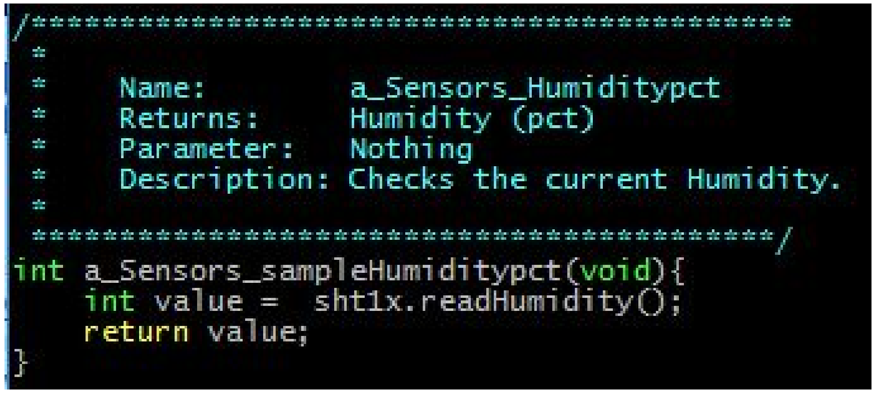
- Uses external libraries to collect data
- Scales the data (scaling ratios from previous Apple Code)
- Functions separated by generations
  - Assigned in configuration



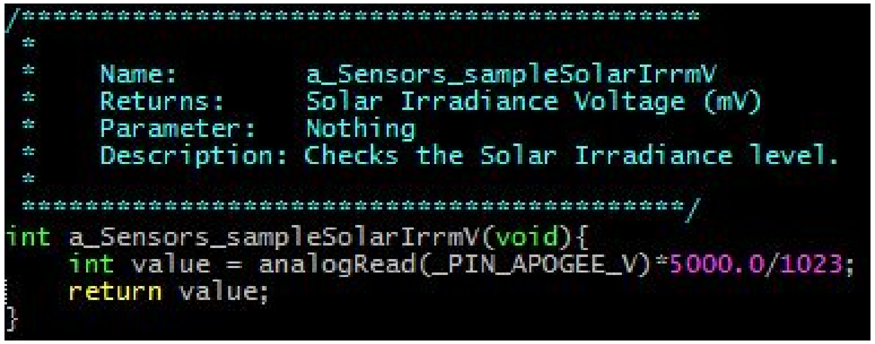
Sensors



```
/*********************************************
 *
 *    Name:        a_Sensors_samplePressurepa
 *    Returns:     Weather Pressure (pa)
 *    Parameter:   Nothing
 *    Description: Checks the current Pressure.
 *
 *********************************************/
int a_Sensors_samplePressurepa(void){
    int value = bmp085.readPressure();
    return value;
}
```



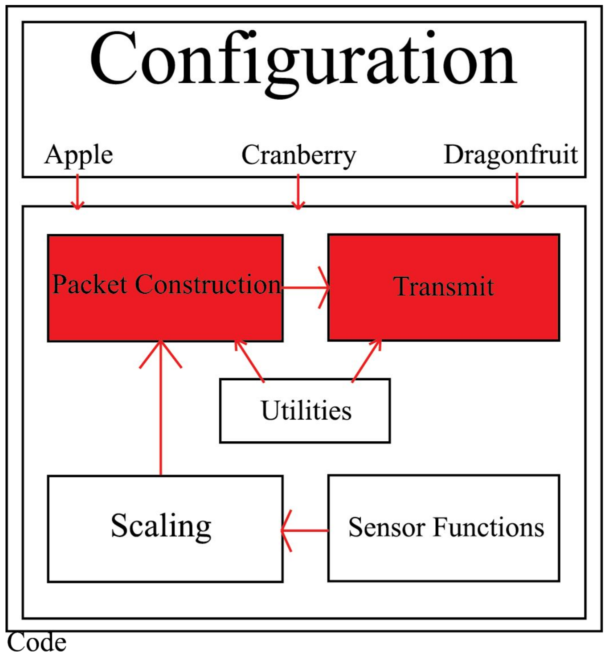
```
/*********************************************
 *
 *    Name:        a_Sensors_Humiditypct
 *    Returns:     Humidity (pct)
 *    Parameter:   Nothing
 *    Description: Checks the current Humidity.
 *
 *********************************************/
int a_Sensors_sampleHumiditypct(void){
    int value =  sht1x.readHumidity();
    return value;
}
```



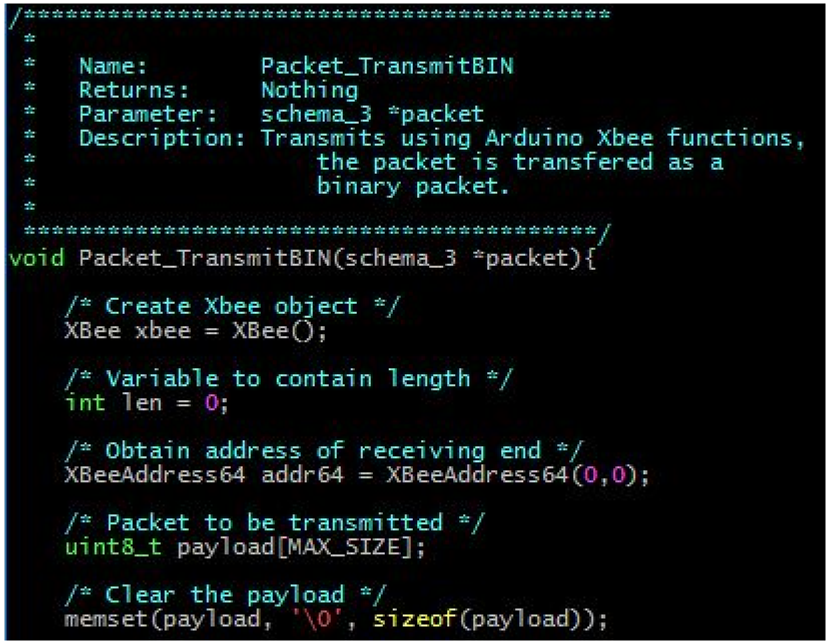
```
/*********************************************
 *
 *    Name:        a_Sensors_sampleSolarIrrmV
 *    Returns:     Solar Irradiance Voltage (mV)
 *    Parameter:   Nothing
 *    Description: Checks the Solar Irradiance level.
 *
 *********************************************/
int a_Sensors_sampleSolarIrrmV(void){
    int value = analogRead(_PIN_APOGEE_V)*5000.0/1023;
    return value;
}
```

Transmit

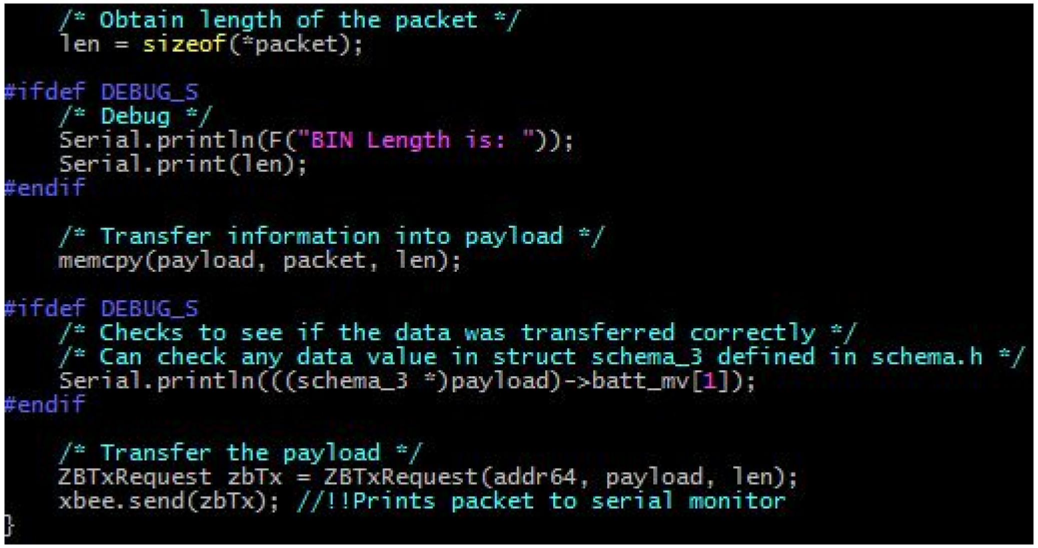
- Applies XBee library
- Two methods of transmitting
  - Binary
    - Struct modeled off of Apple's schema_3
  - UART
- Components:
  - Clear/Initialization
  - Construction
  - Transmit
  - Test Packet Generators



Transmit



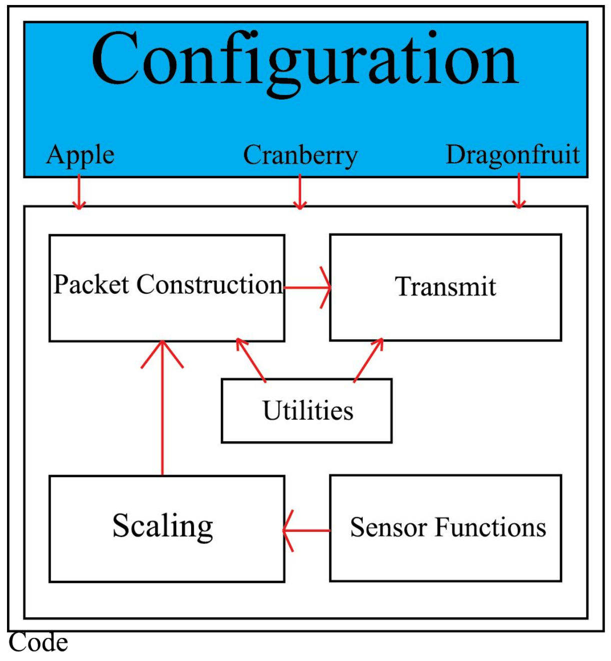
```
/******************************************
 *
 *   Name:        Packet_TransmitBIN
 *   Returns:     Nothing
 *   Parameter:   schema_3 *packet
 *   Description: Transmits using Arduino Xbee functions,
 *                the packet is transfered as a
 *                binary packet.
 *
 ******************************************/
void Packet_TransmitBIN(schema_3 *packet){

    /* Create Xbee object */
    XBee xbee = XBee();

    /* Variable to contain length */
    int len = 0;

    /* Obtain address of receiving end */
    XBeeAddress64 addr64 = XBeeAddress64(0,0);

    /* Packet to be transmitted */
    uint8_t payload[MAX_SIZE];

    /* Clear the payload */
    memset(payload, '\0', sizeof(payload));
```



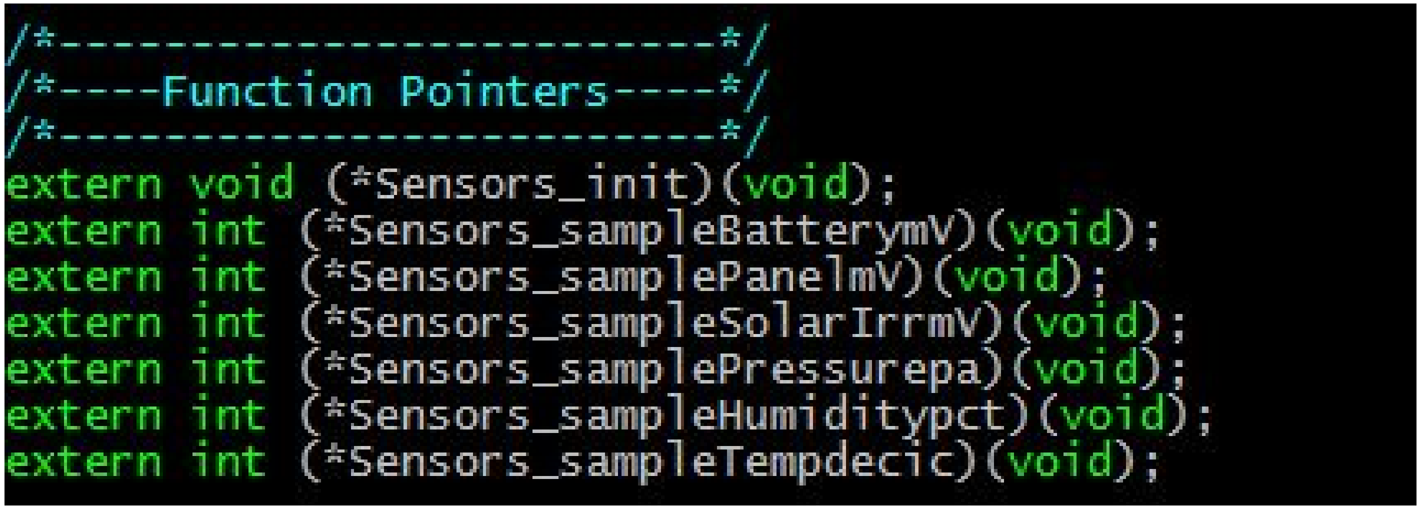
```
    /* Obtain length of the packet */
    len = sizeof(*packet);

#ifdef DEBUG_S
    /* Debug */
    Serial.println(F("BIN Length is: "));
    Serial.print(len);
#endif

    /* Transfer information into payload */
    memcpy(payload, packet, len);

#ifdef DEBUG_S
    /* Checks to see if the data was transferred correctly */
    /* Can check any data value in struct schema_3 defined in schema.h */
    Serial.println(((schema_3 *)payload)->batt_mv[1]);
#endif

    /* Transfer the payload */
    ZBTxRequest zbTx = ZBTxRequest(addr64, payload, len);
    xbee.send(zbTx); //!!Prints packet to serial monitor
}
```

Configuration

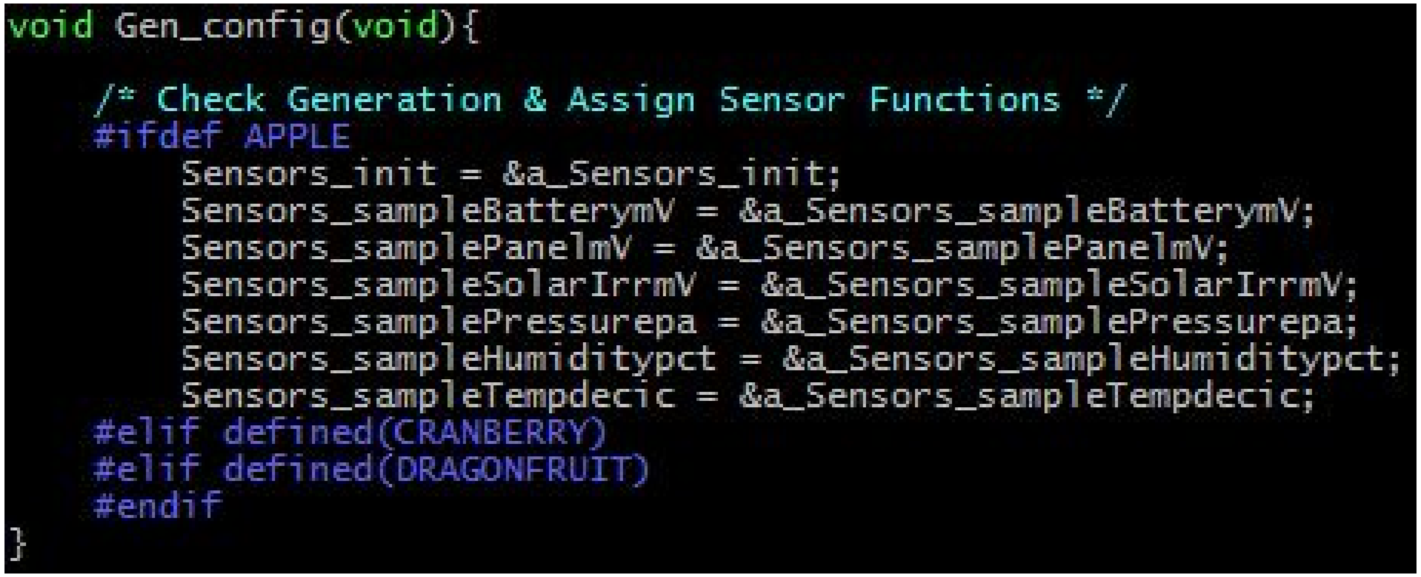
- Function pointer implementation for Sensors
- Pin configurations
- Generation declaration



Configuration



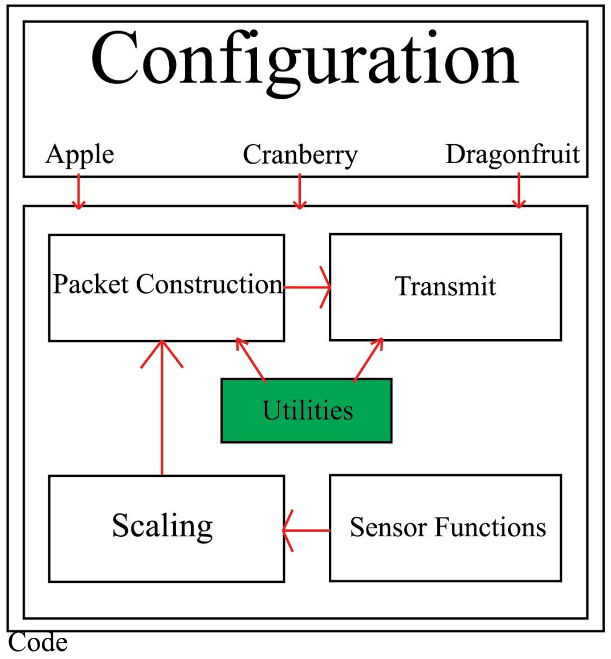
```
/*---------------------------*/
/*----Function Pointers----*/
/*---------------------------*/
extern void (*Sensors_init)(void);
extern int (*Sensors_sampleBatterymV)(void);
extern int (*Sensors_samplePanelmV)(void);
extern int (*Sensors_sampleSolarIrrmV)(void);
extern int (*Sensors_samplePressurepa)(void);
extern int (*Sensors_sampleHumiditypct)(void);
extern int (*Sensors_sampleTempdecic)(void);
```



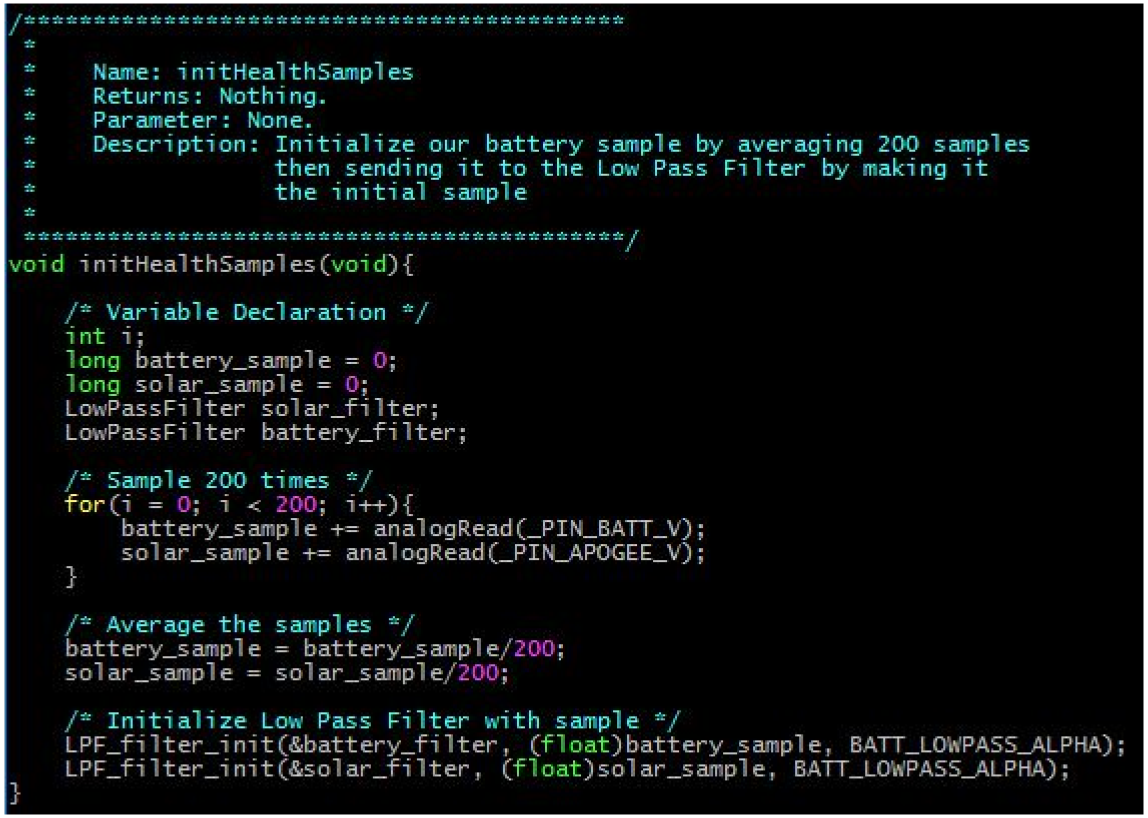
```
void Gen_config(void){

    /* Check Generation & Assign Sensor Functions */
    #ifdef APPLE
        Sensors_init = &a_Sensors_init;
        Sensors_sampleBatterymV = &a_Sensors_sampleBatterymV;
        Sensors_samplePanelmV = &a_Sensors_samplePanelmV;
        Sensors_sampleSolarIrrmV = &a_Sensors_sampleSolarIrrmV;
        Sensors_samplePressurepa = &a_Sensors_samplePressurepa;
        Sensors_sampleHumiditypct = &a_Sensors_sampleHumiditypct;
        Sensors_sampleTempdecic = &a_Sensors_sampleTempdecic;
    #elif defined(CRANBERRY)
    #elif defined(DRAGONFRUIT)
    #endif
}
```

Utilities

- Components:
  - Health check
  - Power management
  - Overflow checker
  - Macro definitions

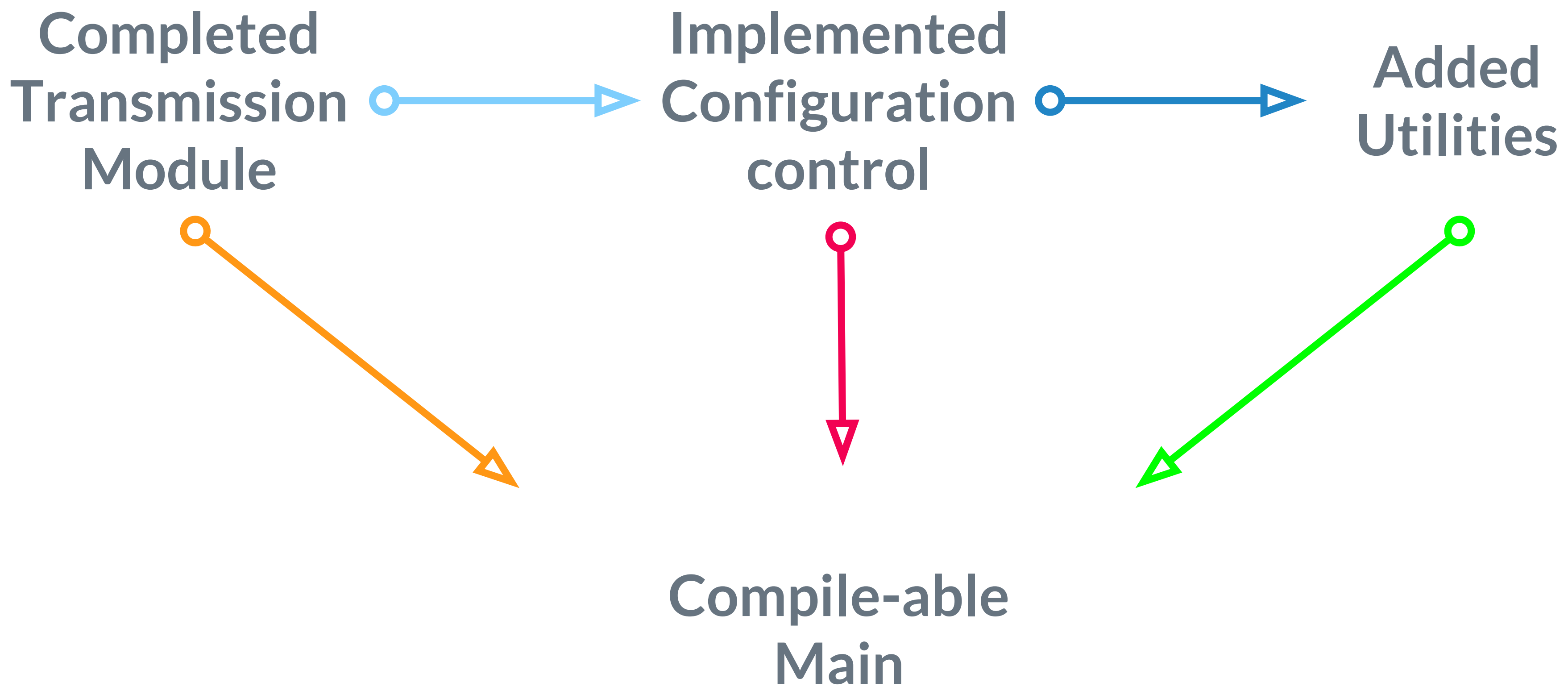


Utilities



```
/*******************************************
 *
 *    Name: initHealthSamples
 *    Returns: Nothing.
 *    Parameter: None.
 *    Description: Initialize our battery sample by averaging 200 samples
 *                 then sending it to the Low Pass Filter by making it
 *                 the initial sample
 *
 *******************************************/
void initHealthSamples(void){

    /* Variable Declaration */
    int i;
    long battery_sample = 0;
    long solar_sample = 0;
    LowPassFilter solar_filter;
    LowPassFilter battery_filter;

    /* Sample 200 times */
    for(i = 0; i < 200; i++){
        battery_sample += analogRead(_PIN_BATT_V);
        solar_sample += analogRead(_PIN_APOGEE_V);
    }

    /* Average the samples */
    battery_sample = battery_sample/200;
    solar_sample = solar_sample/200;

    /* Initialize Low Pass Filter with sample */
    LPF_filter_init(&battery_filter, (float)battery_sample, BATT_LOWPASS_ALPHA);
    LPF_filter_init(&solar_filter, (float)solar_sample, BATT_LOWPASS_ALPHA);
}
```

Progress (from PDR)


Completed Transmission Module
Implemented Configuration control
Added Utilities
Compile-able Main

Remaining Tasks

Utilities

FIx global variable use and documentation

Module Integration

Move configurations now that the modules are working together

Main Code

Clean up and test code

Health check

Implement two routines based on box health

Unit Tests

Test each function independent of hardware, for multiple cases

Poll count

Current code only samples once and then transmits

Thank You!

Any questions?