MicroCART 2013-2014 May 14-10

Kevin Engel, Michael Johnson, Nate Ferris,
William Franey, Kelsey Moore, Lucas Mulkey,
and Aaron Peterson

Scope

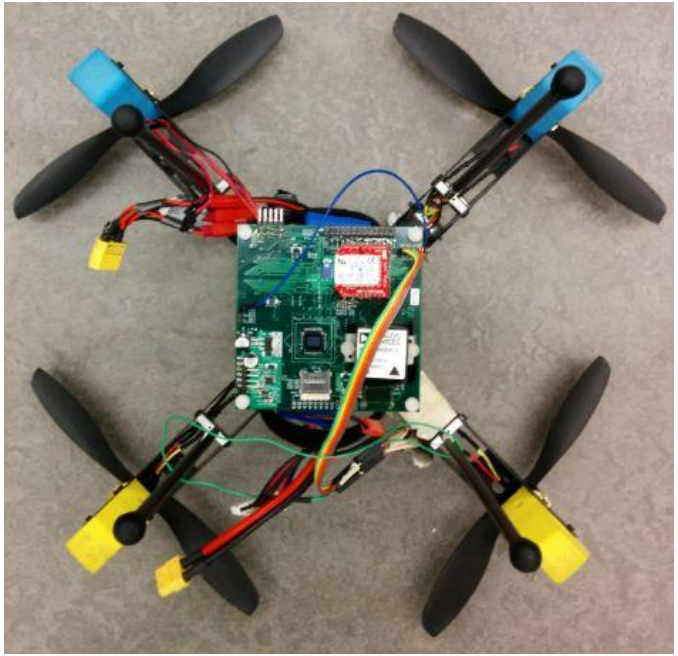
Create on-quad sensor control system consisting of:

Hardware / Sensors:

- Bluetooth
  - basestation commands
- Inertial Measurement Unit (IMU)
  - pitch, roll
- Pseudo GPS
  - lat, lon, alt, yaw

Core Control Programs:

- Proportional, Integral, Derivative (PID) controller
  - analyze current and target location
  - correct error
- Motor interfacing
  - Throttle,Pitch,Roll,Yaw to PWM translation



Design Process

Research: Datasheets, control theory concepts, code examples

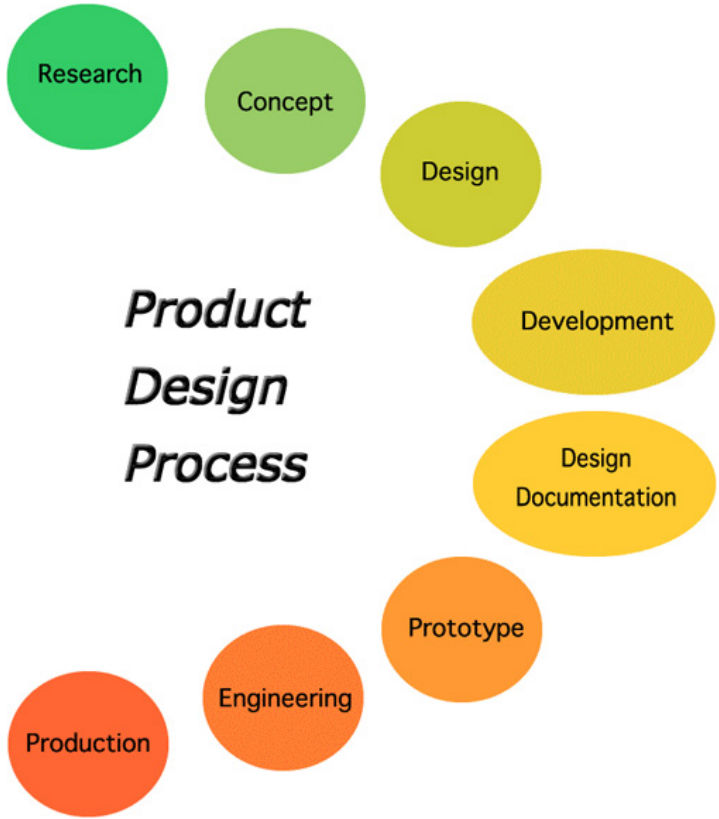
System Design: Developed block diagram detailing data flow and component interaction.

Unit Programs: Created isolated programs dedicated to specific functions (ex. Bluetooth communication, PID controller).

Unit Testing: Tested individual modules for functionality, feasibility and correctness (fix if necessary).

Program Integration: Integrated each unit module into the complete system.

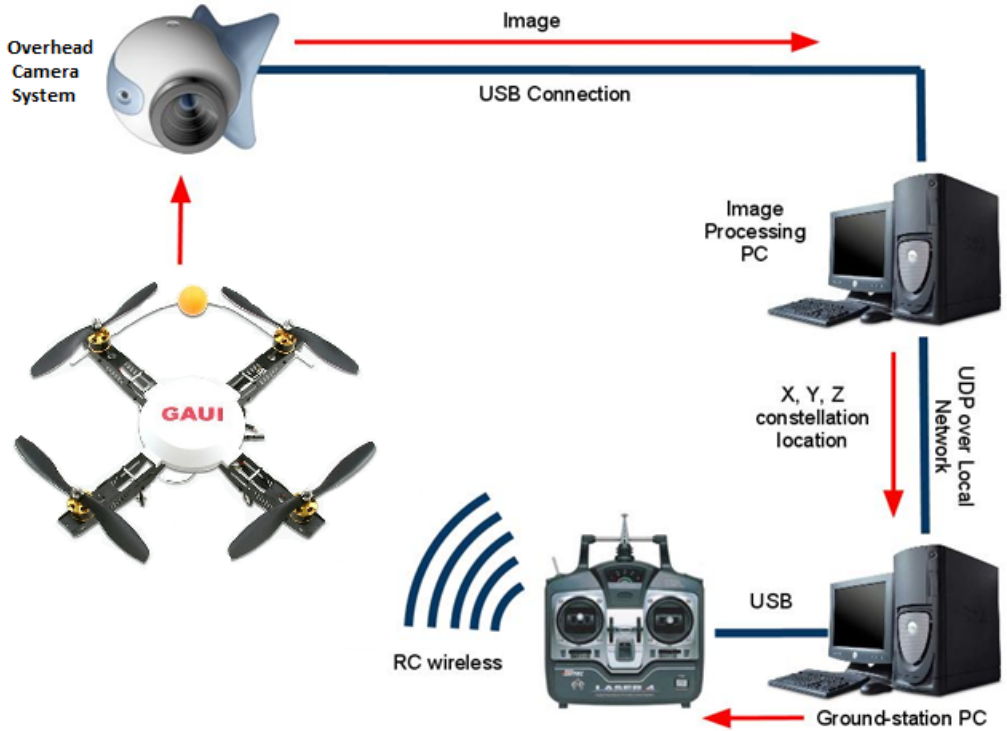
Integration Testing: Tested integrated system for correctness, and that it meets all design requirements.



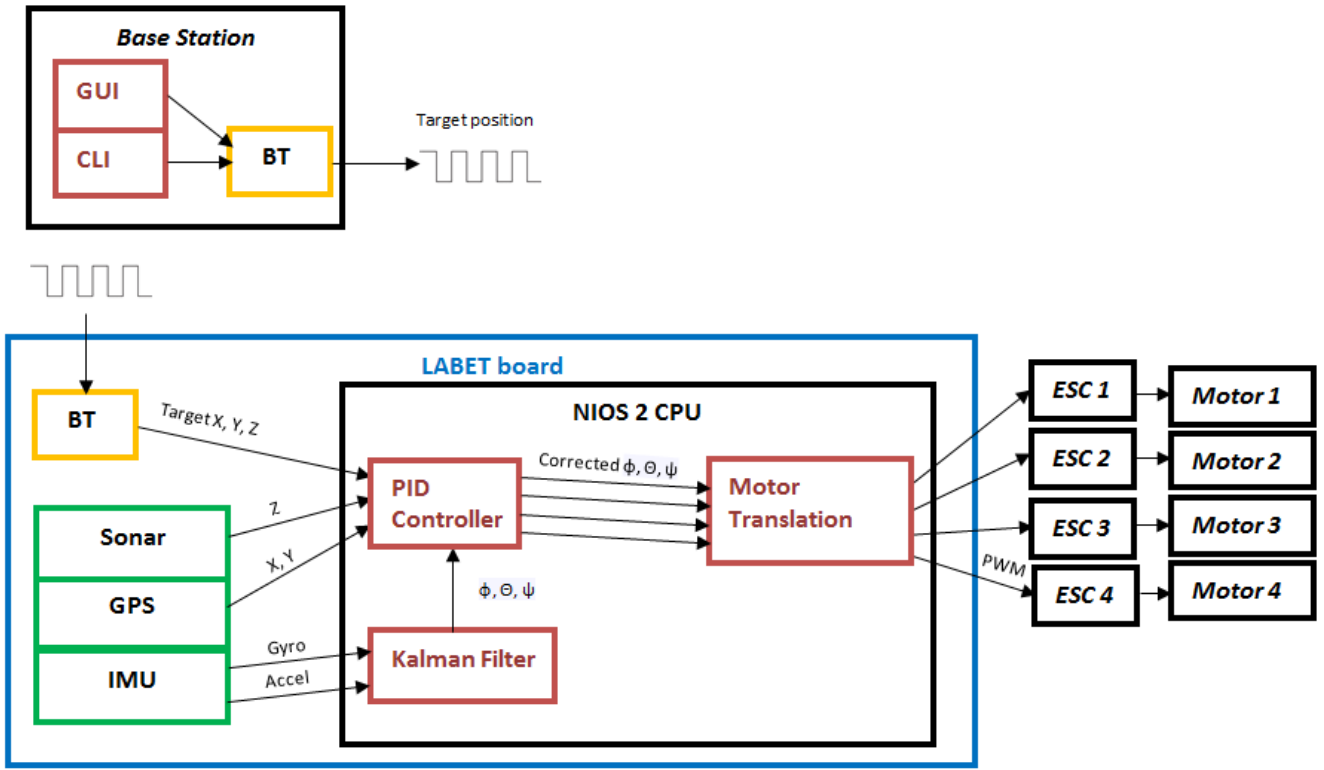
Research

- Ardupilot (MultiWii)
  - Motor mixing
  - IMU Filter, usage
  - GPS usage
  - PID controller
  - Commands
- NMEA 0183
  - GPS sentence protocol
- VRPN
  - IR camera data protocol
- Sensor Datasheets
  - properties
    - max speed, voltage, precision
  - pin outs
  - data protocol
- Other Quads (Hobbyist)
  - Manual flight tutorial
  - Forums, parts & batteries used
  - Recommended care/use
  - Frequent behavior
  - Troubleshooting

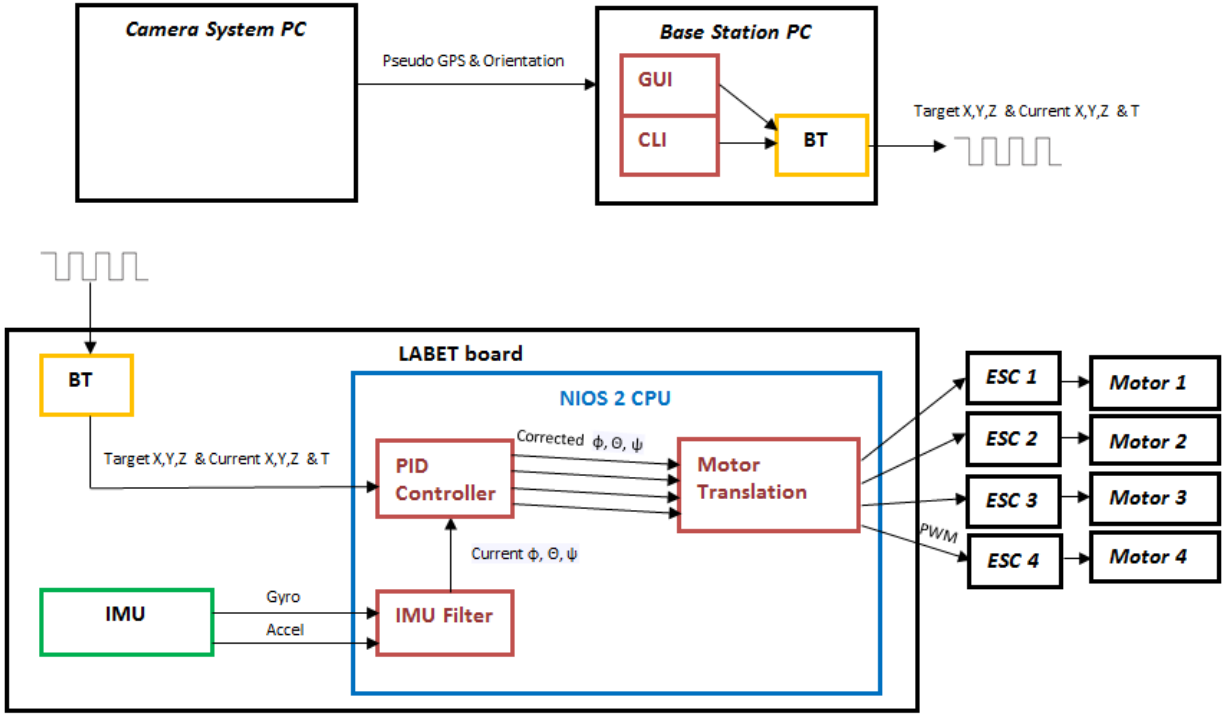
Old System


Overhead
Camera
System
Image
USB Connection
Image
Processing
PC
GAUI
X, Y, Z
constellation
location
UDP over Local
Network
RC wireless
USB
Ground-station PC

Original Design



Current System



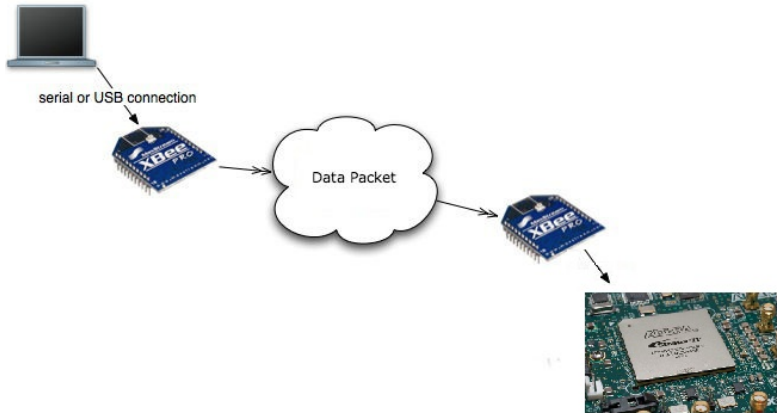
Bluetooth

Task:

- Complete rip and replacement of the old communication system.
- Flight logic, stability, to be ran onboard, rather than on the base station computer.

Approach:

- Test programs to collect data and transfer offboard
- Begin by using all component of the old system (cameras, etc.) EXCEPT, instead of the PPM and receiver, use the bluetooth modules and the servos to output PWM directly to the motors.
- Once successful, we began to remove other remnants of the old system over (Camera system orientation data).



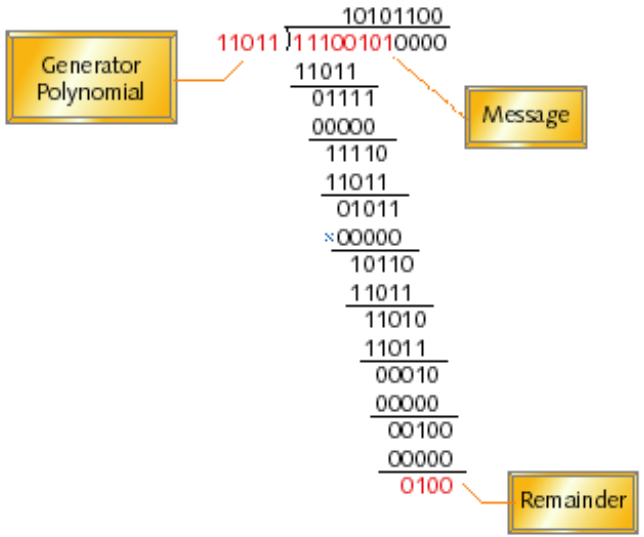
Bluetooth

Complications:

- Unreliable Data Transfers
  - Broken Packets
  - One broken packet can cause the whole system to act unpredictably, completely breaking our flow, and eventually compounding errors.

Solution:

- Find the best baud rate and frequency to maximize speed and minimize transmission errors.
- Start byte for flow control
- Implement Cyclic Redundancy Checks (CRC) to ensure data packet integrity on both sides.



Bluetooth

Results:

- Bluetooth communication has been largely successful beyond the unexplained limitations of our hardware (unable to reliably transmit at the highest speeds).

- Camera position and orientation information can be communicated to the FPGA on-board the vehicle to be used in the PID and adjust the signals going to the motors.

Inertial Measurement Unit (IMU)

Observation:

- In initial tests, quad is not stable using exclusively data from the camera sent over bluetooth
- Hypothesis: The delay between packets is too great for roll and pitch estimations to keep the quad stable

Task:

- Move calculation of roll and pitch onboard using the IMU. This should allow the quad to be stable without bluetooth packets.

IMU

Approach:

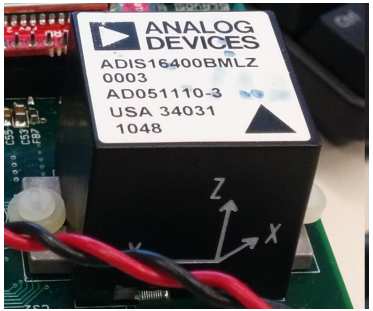
- Implement a simple, lightweight filter to combine accelerometer and gyroscope data and estimate roll and pitch onboard
- These roll/pitch estimations can be updated ~100 times per second, while simultaneously accepting lat/lng/altitude values from the camera system 4 times/sec (pseudo-GPS)

Complications:

- Conductive bottom; shorts board
- Board will not accept threading

Solutions:

- PID constants will need to be retuned, the i term will likely be eliminated for the duration of testing
- Non-blocking read to get camera data, not ideal
- Estimate distance travelled between updates



Kalman Filter

Tasks:

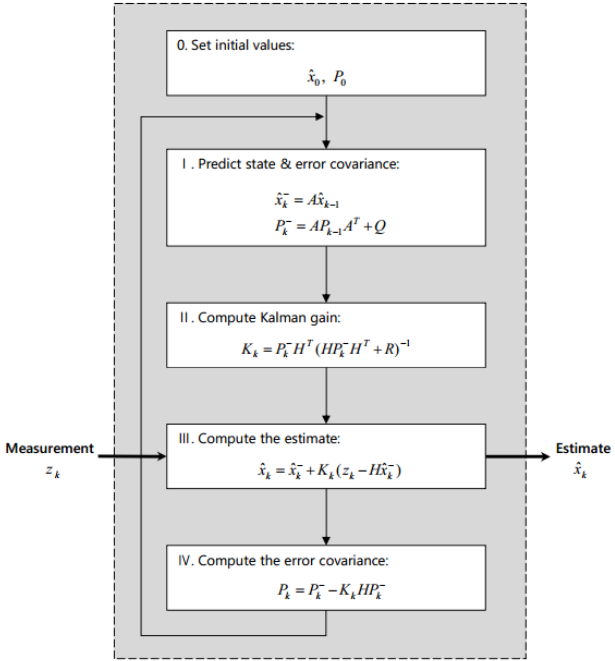
- Get the most accurate approximation of pitch and roll from IMU possible
  - Gyroscopes and accelerometers inherently have their own shortcomings.

Approach:

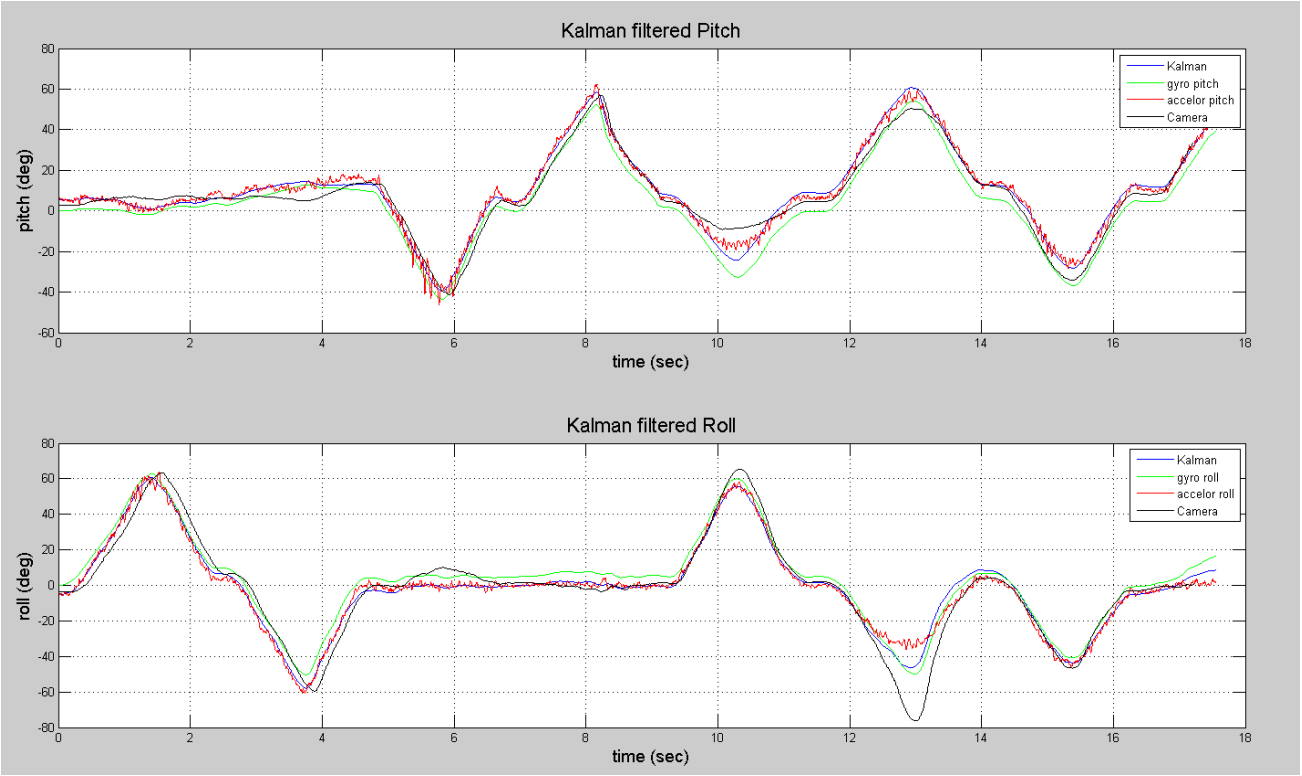
- Write code in Matlab and test by comparing data from our imu sensors with data from the high-speed camera system
- Translate code into C for on-board calculating

Complications

- Kalman filter seemed too heavy for the scope of our project

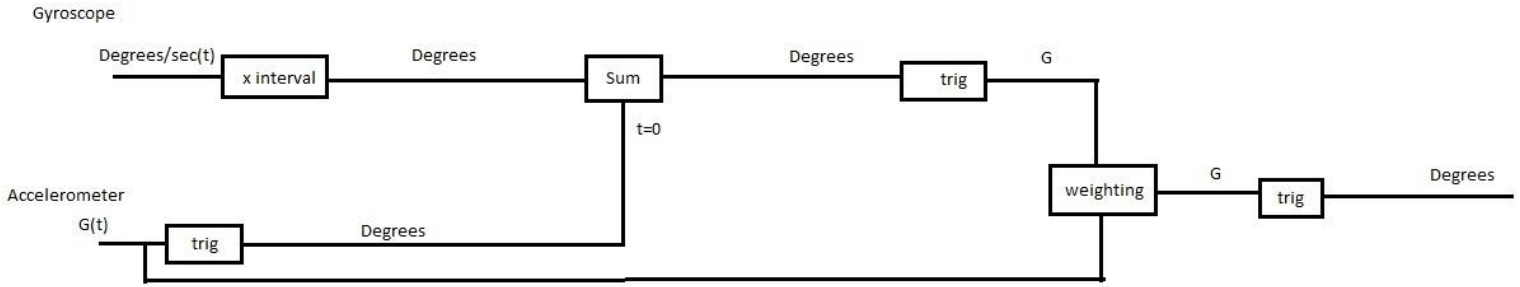


Kalman Filter vs IR Camera



IMU-Lightweight filter

- Based on Kalman filter
- Calculate orientation of quad from two separate sources

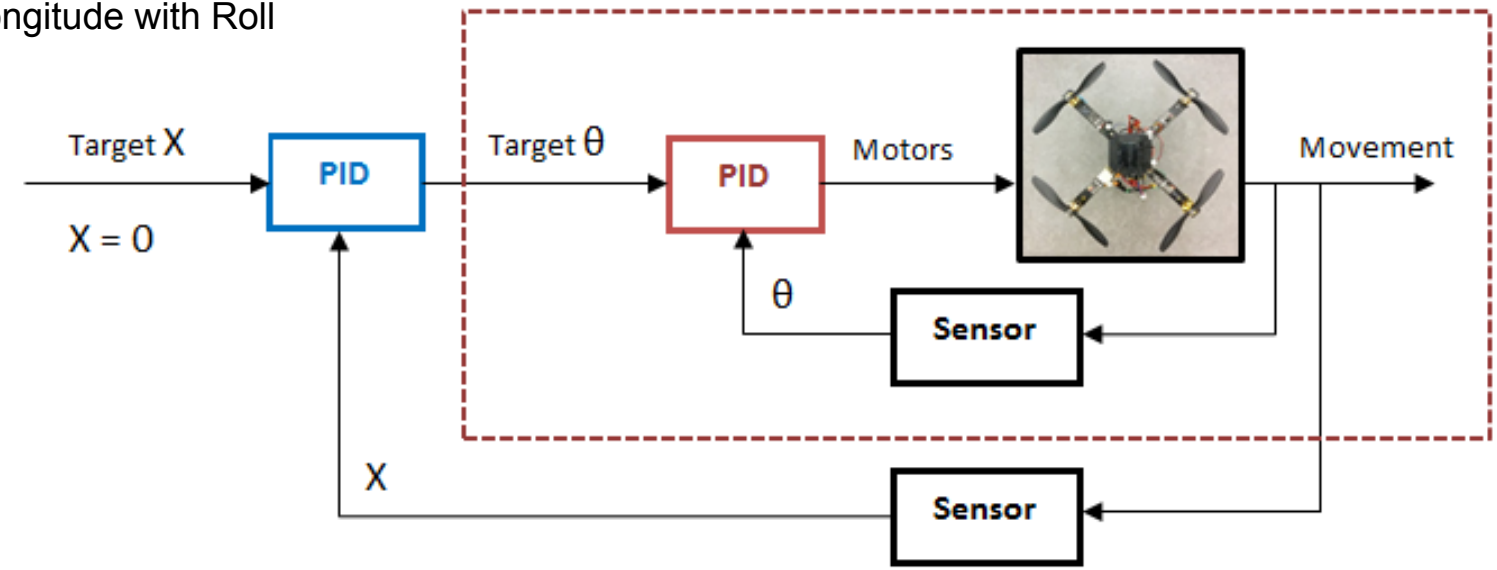


PID Controller

- Task: Quad needs a way to control its movements in flight with minimal data from the Base Station
  - Current orientation and location used to move to new location or stay in place
  - Error calculation based on current and target location
  - Motor correction changed based on error
- Approach: Created new onboard PID controller
  - Nested PID controller - (position (orientation))
  - IMU data integration
  - Lesser camera data for outer loop
- Complications: Timing and Sign errors
  - The timing for the pseudo-GPS was 5 times slower than the data we would receive from the IMU
  - Tuning with new weight and no assistance proved to be very difficult

PID Controller (nested)

Latitude with Pitch
Longitude with Roll



Motor Signal-Mixer

- Replaces the GU344
- Adjusts PWM for each motor
- Output based on 0% to 100%
  - Throttle - 0%
  - Pitch, roll, yaw - 50%
- Throttle ±(Pitch-50)±(Roll-50)±(Yaw-50)
- Flight controller is either added or subtracted

User Interface

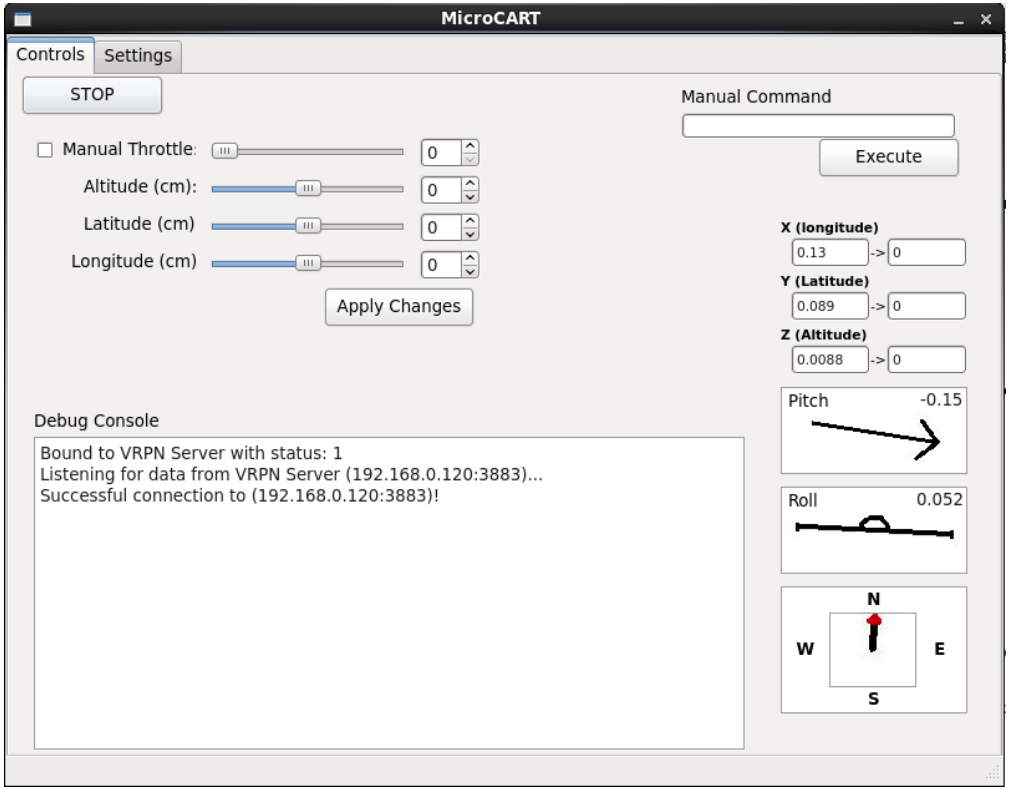
Task: Write a UI to support our new system's components. Ensure the capability of automatic and manual control is preserved through the new protocols.

Approach: Take the UI skeleton from last year's code but convert each class to tailor to our system.

Complications:

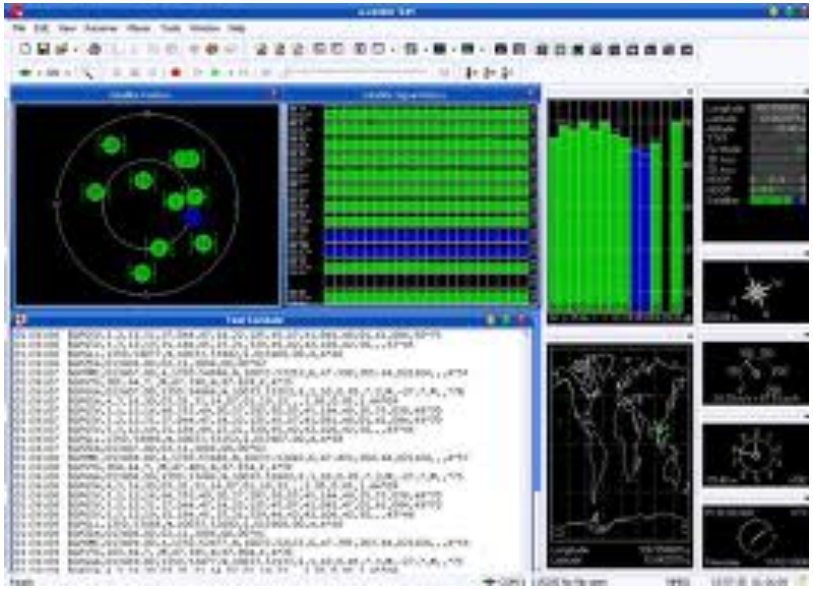
- Modularity.
- Concurrency

Results: A UI with the look and feel of the old system, but with the internals of the new system.



GPS - hardware

- Seeed’s GPSbee
- U-Center
  - baud rate
  - update rate
  - output messages



GPS - software

- NMEA 0183
  - National Marine Electronics Association
  - communication protocol
  - example:

$GPGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,x.x,M,x.x,M,x.x,xxxx<cr><lf>

- C code:
  - GGA - lat, lon, quality, alt
  - locate start, tokenize, parse

```
while(1)
{
    //read GPS sentence, form string, and notify tokenizer
    if(read(fdGPS, &readBuffer, 1))
    {
        //located beginning of GPS setence
        if(readBuffer == '$')
        {
            isReadingGPS = 1;
            hasGPS = 0;
            memset(GPSsentence, 0, SIZE_GPS);
            /*
            //test
            printf("found beginning\n");
            */
        }

        //located end of GPS sentence
        if(readBuffer == '\r')
        {
            isReadingGPS = 0;
            hasGPS = 1;
            /*
            //test
            printf("found end\n");
            */
            printf("%s\n", GPSsentence);
        }

        //forming string of GPS setence
        if(isReadingGPS && readBuffer != '$' && readBuffer != '\r' &&
        {
            strAppendChar(GPSsentence, readBuffer);
        }
    }

    //tokenize GPS sentence, parse and store important data
    if(hasGPS)
    {
        GPStoken = strtok(GPSsentence, delimiters);
        tokenCount++;

        if(strcmp(GPStoken, "GPGGA") == 0)
        {
            while(GPStoken != NULL)
```

Other Technical Challenges

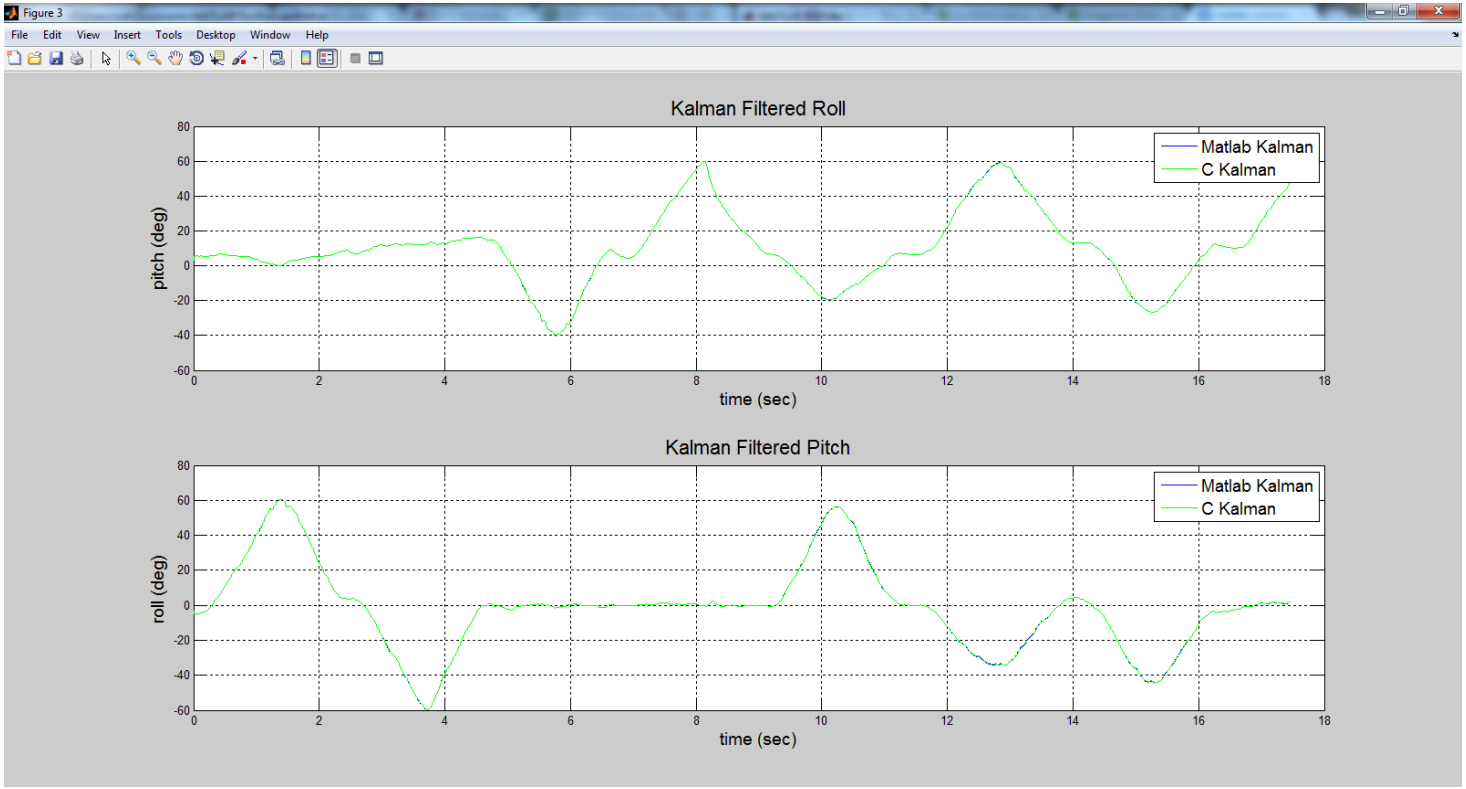
- Bad Motors
  - years of use
  - overall durable yet constant flight tests led to shortened life
- Old, custom PCB Board
  - relatively slow processor
  - problems with BT communication, sensor processing
  - insufficient number of standard port hookups causes messy wiring
  - Inability to implement threading
- Quadcopter Frame
  - forces board to be placed offset from center of mass, same with batteries
  - Plastic “snaps” apart frame occasionally
  - Foam bumpers alter quad model, PID constants

Questions?

Kalman Filter Conversion

- Task:
  - take current matlab code file(s) and convert them into a C file suitable for running
- Approach:
  - added math.h in order to be able to run sine, cosine, atan2 etc. From there continue syntactical math calculations, create functions
- Complications:
  - Matlab is considered a “calculator” -- faced difficulties in terms of math calculations
  - Simple matrix algebra became complicated, able to overcome this by adding an additional matrix_util
- Results:
  - created Add, Subtract, Multiply, Transpose, and Inverse functions for the matrix aimed file. Code was written for “All” matrix sizes, however we only used 4x4 and 4x1. Compiles and runs perfectly.

C Kalman vs Matlab Kalman



Kalman Filter

Complications:

- Complexity of the Kalman filter led to difficulty in debugging
- Uncertain if we can trust our “true” Camera values
- Filter seemed to heavy

Solution:

- Strong technical communication with advisors and CprE’s
- Light-weight filter

```
// outputOrientation is an array of pitch roll yaw values
// dt is delta time
// p,q,r are the gyroscope outputs
// ax,ay,az are the accelerometer outputs
void Kalman_getOrientation(float* outputOrientation, float dt, flo
    float z[4][1];
    float A[4][4];
    float theta_a = 0;
    float phi_a = 0;
    float psi_a = 0;
    float phi = 0;
    float theta = 0;
    float psi = 0;

    dt = dt*.001;
    p = p*3.1415/180;
    q = q*3.1415/180;
    r = r*3.1415/180;
    if(initializeFlag == 0)
    {
        Kalman_init();
        initializeFlag = 1;
    }
//setup data for Euler Kalman algorithm
    Kalman_calcA(A, dt, p, q, r);
    Kalman_eulerAccel(&phi_a, &theta_a, ax, ay, az); //done
    Kalman_eulerToQuaternion(z, phi_a, theta_a, psi_a);  //done

//perform Kalman filter algorithm
    Kalman_eulerKalman(&phi, &theta, &psi, A, z);

    outputOrientation[0] = phi;
    outputOrientation[1] = theta;
    outputOrientation[2] = psi;

}
```

GPS - task

Task:

- Obtain current x,y,z position of quad through GPS

Approach:

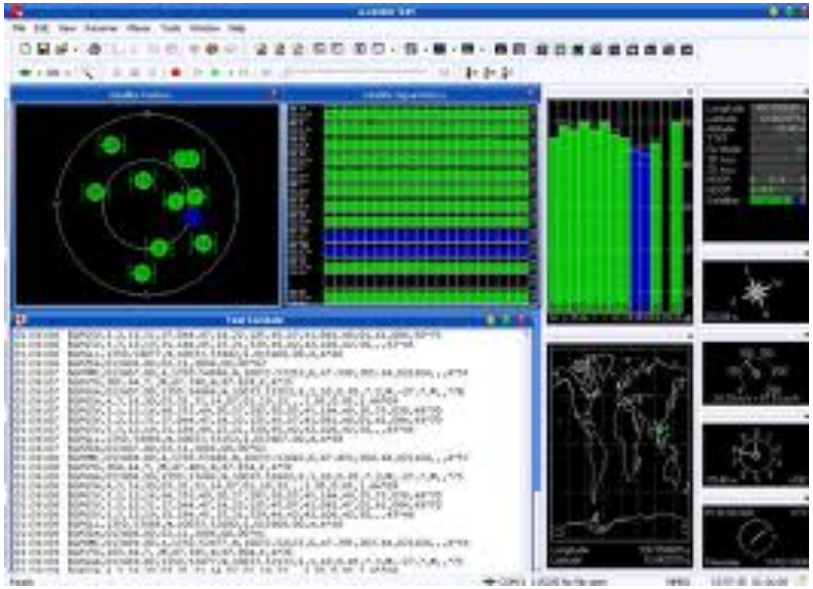
- research and compare GPS hardware specs
- research message protocol
- configure hardware using associated software tool
- create C code to parse messages and pick out essential information

Complications:

- Non-volatile memory was unable to be configured using software tool.
- Not always locked - code should handle this case
- Inaccurate - drifting location
- Imprecise - differential vs non-differential fix

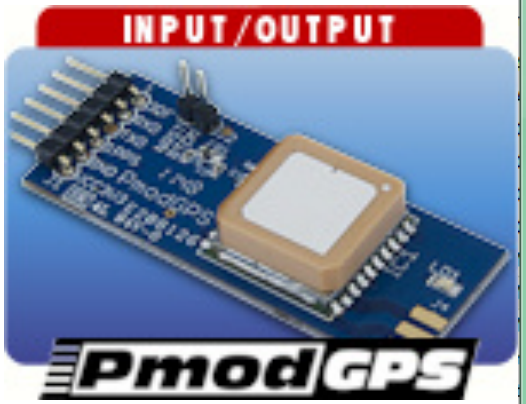
GPS - hardware

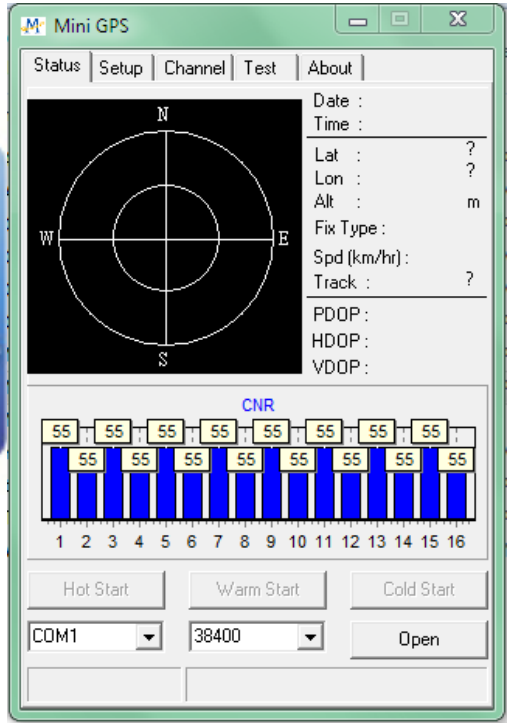
- Seeed's GPSbee
  - 4Hz update
  - Regular Fix
    - meters
- U-Center
  - baud rate
  - update rate
  - output messages



GPS - hardware

- Digilent's PmodGPS
  - 10Hz update
  - Differential Fix
    - centimeters
- MiniGPS
  - baud rate
  - update rate
  - output messages





PID Constants

Task:

- Attachment of the labet board, legs, and bumpers to the quad changed its center of mass and our PID controller no longer functions properly
- Get accurate initial PID constants that we can tune our PID controller with

Procedure:

- Create CAD models of the additional quad components
- Import the components into Matlab SimMechanics and attach them to an existing CAD model of the quad
- Setup the simulation environment to obtain rudimentary PID constants

Complications:

- Limited knowledge of both SolidWorks and SimMechanics made the CAD import very tricky
  - Complexity of the quad made a SimMechanics assembly unfeasible
  - CAD assemblies failed to import some components