Quadcopter

3.2

Generated by Doxygen 1.8.16

Chapter 1

Quadcopter

1.1 Description

The C++ Arduino firmware for a quadcopter made as part of UCSD class CSE 176e

1.2 Hardware

Custom PCB utilizing the ATmega128RFA1.

- Quadcopter hardware by Thomas Stuart and Robert Ketchum. Documentation and EAGLE board files available here: https://github.com/rketch/quadcopter
- Remote PCB made by UCSD Professor Steven Swanson.

1.3 Libraries

All libraries available at: https://github.com/rketch/quadcopter

- Wire
  - Arduino I2C
- Arduino_LSM9DS1
  - Onboard IMU for quadcopter orientation.
- Adafruit_Simple_AHRS
  - Attitude and heading reference system for LSM
- radio
  - Radio library. Modified to include data structure sent via radio
- quad_remote
  - Custom CSE 176e library for the remote
- EEPROM
  - To save PID and trim calibration values to Non-volatile memory
- RotaryEncoder
  - To use the knob and button

1.4 Authors

Created by Thomas Stuart and Robert Ketchum, April 2019. Modified by Robert Ketchum, May 2021.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

File Documentation

3.1 quad_firmware/quad_firmware.ino File Reference

```
#include "radio.h"
#include <Wire.h>
#include <Adafruit_Simple_AHRS.h>
#include <Adafruit_LSM9DS1.h>
```

Macros

- #define RADIO_CH 17

  Radio Channel.
- #define M1 35

  Pin 35 is mapped to motor 1.
- #define M2 34

  Pin 34 is mapped to motor 2.
- #define M3 8

  Pin 8 is mapped to motor 3.
- #define M4 9

  Pin 9 is mapped to motor 4.
- #define FILT_RATIO 0.91

  Complementary filter ratio for a time constant of 10 Hz = 0.91.
- #define MS_TO_S 1000

  Conversion ratio.
- #define NEUTRAL_PITCH 43

  Neutral pitch gimbal position.
- #define NEUTRAL_ROLL 60

  Neutral roll gimbal position.
- #define NEUTRAL_YAW 60

  Neutral yaw gimbal position.
- #define MAX_ANGLE_FROM_NEUTRAL 10

  Maximum pitch or roll from neutral.
- #define MAX_TRIM_ANGLE 30

  Maximum pitch or roll trim angle.
- #define YAW_MAX_ANG_VEL 60

  Maximum yaw angular velocity (deg/sec)
- #define TRIM_HISTORY 3

  Number of median filtered trim values.

Functions

- Adafruit_Simple_AHRS ahrs & lsm (), &lsm.getMag(), &lsm.getGyro()
- void setupLSM ()
- void setup ()
- void loop ()
- void verifyRadioData ()
- void ComplimentaryFilter (double dt)
- void PID (double dt)
- void findGimbalOffsets ()
- void findTrimOffsets ()
- float median (float medArray[ ], int n)
- void sort (float unsortedArray[ ], int n)
- void calibrateAccelerometer ()
- void mixer ()
- void initializeMotors ()
- void disableMotors ()
- void engageMotors ()

Variables

- unsigned long time

  Current time since start.
- unsigned int last = millis()

  Last loop time since start.
- double timeDiff = 0.000000

  Implemented in complementary filter and controller.
- struct Data dataFromRemote

  Data structure received over radio.
- bool resetFlag = false

  For zeroing off orientation offset when armed.
- quad_data_t orientation

  Data structure with quadcopter orientation.
- float accHistoryArr [3][2]

  pitch, roll accelerometer reading history
- float accHistoryArrCopy [3]

  For median filter.
- float rawGyro [2]

  Raw gyro pitch, roll rate.
- float gyroAngle [2]

  Integrated gyro angles.
- float rawAcc [2]

  Raw accelerometer angles.
- float filteredAngle [2]

  Complementary filtered angle.
- float filtGyroAngle [2]

  Partially filtered gyro angle.
- float filtAccelAngle [2]

  Partially filtered accelerometer angle.
- float gyroAngleStepBack [2]

  Previous time step filtered gyroscope angle. Needed for complementary filter.

- float angleOffset [2]

  Accelerometer offset from neutral. Measured when armed.
- float trimAngle [2]

  Trimm offsets.
- float trimHistoryArr [3][4]

  Pitch, roll trim floats.
- float trimHistoryArrCopy [3]

  Trim History for median filter.
- float medOutTrimArray [4]

  Pitch, roll median filter outputs.
- float PID_output [3]

  PID output.
- float remoteSetAngle [3]

  Set angle.
- float currentAngle [3]

  Current quadcopter angle.
- float error [3]

  Error angle.
- float errorSum [3]

  Integrated error.
- float lastError [3]

  previous error (for derivative)
- float MotorValue [4]

  Implemented motor thrust variables.
- Adafruit_LSM9DS1 lsm = Adafruit_LSM9DS1()

  Create LSM9DS1 board instance.

3.1.1 Macro Definition Documentation

3.1.1.1 FILT_RATIO

```
#define FILT_RATIO 0.91
```

Complementary filter ratio for a time constant of 10 Hz = 0.91.

3.1.1.2 M1

```
#define M1 35
```

Pin 35 is mapped to motor 1.

3.1.1.3 M2

```
#define M2 34
```

Pin 34 is mapped to motor 2.

3.1.1.4 M3

```
#define M3 8
```

Pin 8 is mapped to motor 3.

3.1.1.5 M4

```
#define M4 9
```

Pin 9 is mapped to motor 4.

3.1.1.6 MAX_ANGLE_FROM_NEUTRAL

```
#define MAX_ANGLE_FROM_NEUTRAL 10
```

Maximum pitch or roll from neutral.

3.1.1.7 MAX_TRIM_ANGLE

```
#define MAX_TRIM_ANGLE 30
```

Maximum pitch or roll trim angle.

3.1.1.8 MS_TO_S

```
#define MS_TO_S 1000
```

Conversion ratio.

3.1.1.9 NEUTRAL_PITCH

```
#define NEUTRAL_PITCH 43
```

Neutral pitch gimbal position.

3.1.1.10 NEUTRAL_ROLL

```
#define NEUTRAL_ROLL 60
```

Neutral roll gimbal position.

3.1.1.11 NEUTRAL_YAW

```
#define NEUTRAL_YAW 60
```

Neutral yaw gimbal position.

3.1.1.12 RADIO_CH

```
#define RADIO_CH 17
```

Radio Channel.

3.1.1.13 TRIM_HISTORY

```
#define TRIM_HISTORY 3
```

Number of median filtered trim values.

3.1.1.14 YAW_MAX_ANG_VEL

```
#define YAW_MAX_ANG_VEL 60
```

Maximum yaw angular velocity (deg/sec)

3.1.2 Function Documentation

3.1.2.1 calibrateAccelerometer()

```
void calibrateAccelerometer ( )
```

Calculate the angle offset of the accelerometer

3.1.2.2 ComplimentaryFilter()

```
void ComplimentaryFilter (
            double dt )
```

Combine the gyroscope and accelerometer to get an accurate heading of the quadcopter

Parameters

dt	The time difference between calling the function (in ms)

3.1.2.3 disableMotors()

```
void disableMotors ( )
```

Ensure that the motors do not run when the quad is disarmed

3.1.2.4 engageMotors()

```
void engageMotors ( )
```

Write the motor commands to the respective motors

3.1.2.5 findGimbalOffsets()

```
void findGimbalOffsets ( )
```

Find the remote set angles from the gimbal positions

3.1.2.6 findTrimOffsets()

```
void findTrimOffsets ( )
```

Median the trim data and convert to a float angle.

3.1.2.7 initializeMotors()

```
void initializeMotors ( )
```

Initialize motor pins

3.1.2.8 loop()

```
void loop ( )
```

Loop forever. Receive radio data, find current quadcopter orientation, compute control system values, find current throttle corrections,write throttle values to motors if armed.

3.1.2.9 lsm()

```
Adafruit_Simple_AHRS ahrs& lsm ( ) &
```

3.1.2.10 median()

```
float median (
            float medArray[],
            int n )
```

Find the middle value of three raw inputs to filter out pertubations. Thanks geeksforgeeks

Parameters

medArray	An array containing the last n sensed values
n	The length of medArray

Returns

The median value

3.1.2.11 mixer()

```
void mixer ( )
```

Mix the user throttle value and the PID controller output

3.1.2.12 PID()

```
void PID (
            double dt )
```

Utilize PID controller to ensure stable quadcopter flight

Parameters

dt	The time difference between calling the function (in ms)

3.1.2.13 setup()

```
void setup ( )
```

Setup the quadcopter by initializing the radio, motors, IMU registers, and start clock

3.1.2.14 setupLSM()

```
void setupLSM ( )
```

set up our instance of the sensor with the wanted register values

3.1.2.15 sort()

```
void sort (
            float unsortedArray[],
            int n )
```

Sort an array into increasing numerical values. Thanks tsbrownie on youtube.

Parameters

unsortedArray	An array containing the last n sensed values
n	The length of medianFinder

3.1.2.16 verifyRadioData()

```
void verifyRadioData ( )
```

Read radio data received from the remote. If it is verified, save the data in a structure

3.1.3 Variable Documentation

3.1.3.1 accHistoryArr

```
float accHistoryArr[3][2]
```

pitch, roll accelerometer reading history

3.1.3.2 accHistoryArrCopy

```
float accHistoryArrCopy[3]
```

For median filter.

3.1.3.3 angleOffset

```
float angleOffset[2]
```

Accelerometer offset from neutral. Measured when armed.

3.1.3.4 currentAngle

```
float currentAngle[3]
```

Current quadcopter angle.

3.1.3.5 dataFromRemote

```
struct Data dataFromRemote
```

Data structure received over radio.

3.1.3.6 error

```
float error[3]
```

Error angle.

3.1.3.7 errorSum

```
float errorSum[3]
```

Integrated error.

3.1.3.8 filtAccelAngle

```
float filtAccelAngle[2]
```

Partially filtered accelerometer angle.

3.1.3.9 filteredAngle

```
float filteredAngle[2]
```

Complementary filtered angle.

3.1.3.10 filtGyroAngle

```
float filtGyroAngle[2]
```

Partially filtered gyro angle.

3.1.3.11 gyroAngle

```
float gyroAngle[2]
```

Integrated gyro angles.

3.1.3.12 gyroAngleStepBack

```
float gyroAngleStepBack[2]
```

Previous time step filtered gyroscope angle. Needed for complementary filter.

3.1.3.13 last

```
unsigned int last = millis()
```

Last loop time since start.

3.1.3.14 lastError

```
float lastError[3]
```

previous error (for derivative)

3.1.3.15 lsm

```
Adafruit_LSM9DS1 lsm = Adafruit_LSM9DS1()
```

Create LSM9DS1 board instance.

3.1.3.16 medOutTrimArray

```
float medOutTrimArray[4]
```

Pitch, roll median filter outputs.

3.1.3.17 MotorValue

```
float MotorValue[4]
```

Implemented motor thrust variables.

3.1.3.18 orientation

```
quad_data_t orientation
```

Data structure with quadcopter orientation.

3.1.3.19 PID_output

```
float PID_output[3]
```

PID output.

3.1.3.20 rawAcc

```
float rawAcc[2]
```

Raw accelerometer angles.

3.1.3.21 rawGyro

```
float rawGyro[2]
```

Raw gyro pitch, roll rate.

3.1.3.22 remoteSetAngle

```
float remoteSetAngle[3]
```

Set angle.

3.1.3.23 resetFlag

```
bool resetFlag = false
```

For zeroing off orientation offset when armed.

3.1.3.24 time

```
unsigned long time
```

Current time since start.

3.1.3.25 timeDiff

```
double timeDiff = 0.000000
```

Implemented in complementary filter and controller.

3.1.3.26 trimAngle

```
float trimAngle[2]
```

Trimm offsets.

3.1.3.27 trimHistoryArr

```
float trimHistoryArr[3][4]
```

Pitch, roll trim floats.

3.1.3.28 trimHistoryArrCopy

```
float trimHistoryArrCopy[3]
```

Trim History for median filter.

3.2 remote_firmware/remote_firmware.ino File Reference

```
#include "quad_remote.h"
#include "radio.h"
#include <EEPROM.h>
#include <RotaryEncoder.h>
```

Macros

- #define RADIO_CH 17

  Radio Channel.
- #define MAX_TRIM_ANGLE 30

  Maximum trim angle.
- #define TRIM_INCREMENT 10

  Increment by which trim is changed on remote.

Functions

- void knobPressed (bool)
- void knobsUpdate ()
- int calculateColPos (int, bool)
- void updateDisplayPIDElementChange ()
- void converToDecimalsBeforeSend ()
- float castIntPairToDecimal (int, int)
- void readPIDArrayFromEeprom (int)
- void writePIDToEeprom ()
- void writeGimbalsToEeprom ()
- void setup ()
- void loop ()
- void PIDTuningState ()
- void TrimTuningState ()
- void updateDisplayTrimElementChange ()
- void calculateTuningValues ()
- void readAndMapGimbals ()
- void isInArmState (Data &d)
- void calibrate ()
- void updateDisplayPIDNumChange ()
- void eepromRead ()
- void readTrimFromEeprom (int EepromAddress)
- void resetCoefficientValuesAndSigns ()
- void initializeButtonPins ()

Variables

- struct Data remoteToQuadData

  Structure for radio data. Struct "Data" initialized in radio.h.
- int gimbalRawValues [4]

  Values read by gimbal. Nominally 0 - 1024 but constrained by the potentiometer.
- int lowestGimbalValueReadSoFar [4] = { 0, 0, 0, 0}

  lowest value possible by analog stick gimbal
- int largestGimbalValueReadSoFar [4] = {303, 318, 320, 306}

  highest value possible by analog stick gimbal
- int saveLargestGimbalValuesSoFar [12]

  255 + 255 + 255 = 765 is the highest save value with EEPROM for gimbal
- bool knob_btn = 0

  Whether the knob button is pressed.
- bool armToken = true

  State variable for armed mode.
- bool isInTrimMode = false

  State variable for trim mode.
- int cpa [3][6]

  Coefficients Properties Array for displaying and storing PID coefficients for pitch, yaw, roll. 3 rows, 6 columns.
- int sign [9] = {1,1,1, 1,1,1, 1,1,1}

  Sign of PID tuning coefficients for storing via EEPROM.
- int PIDpos = 0

  PID value being tuned currently.
- bool changeLeft = 1

  Keeps track of whether we are tuning whole integer or decimal PID value.
- int row = 0

  LCD screen row to write to.
- int col = 0

  LCD screen column to write to.
- float trimAngle [2]

  Trim pitch and roll angles.
- String top = ""

  Initialize string to print on top row of LCD screen for PID tuning.
- int colPosStart [3]

  What column we are writing to LCD screen for PID tuning.
- bool knob_token = false

  trim or PID tuning variable
- int timesOverflowed = 0

  Keeps track of times an integer has overflowed for EEPROM writing.

3.2.1 Macro Definition Documentation

3.2.1.1 MAX_TRIM_ANGLE

```
#define MAX_TRIM_ANGLE 30
```

Maximum trim angle.

3.2.1.2 RADIO_CH

```
#define RADIO_CH 17
```

Radio Channel.

3.2.1.3 TRIM_INCREMENT

```
#define TRIM_INCREMENT 10
```

Increment by which trim is changed on remote.

3.2.2 Function Documentation

3.2.2.1 calculateColPos()

```
int calculateColPos (
            int pidPos,
            bool tuningWholeInteger )
```

Determines which cpa array column value we wish to edit

Parameters

pidPos	0->2: Pitch, Roll, Yaw
tuningWholeInteger	Whether we are tuning an integer or decimal

3.2.2.2 calculateTuningValues()

```
void calculateTuningValues ( )
```

Calculates the tuning values displayed and sent to the quad (float) from the values saved to EEPROM (int)

3.2.2.3 calibrate()

```
void calibrate ( )
```

Calibrates the gimbals and call function writeGimbalsToEeprom to save the new values. Scripted and hardcoded out of necessity.

3.2.2.4 castIntPairToDecimal()

```
float castIntPairToDecimal (
            int left_int,
            int right_int )
```

Takes the separated PID integers displayed on the LCD and combines them into one float value

Parameters

left_int	integer in the ones place on the LCD screen
right_int	integer in the decimal place on the LCD screen

Returns

castedPIDValue float value which may be used in computation

3.2.2.5 converToDecimalsBeforeSend()

```
void converToDecimalsBeforeSend ( )
```

Takes the PID user input as displayed on the LCD (as unsigned integers) and saves it in the structure "remoteTo↩ QuadData" (as floats), which will be sent over radio to the quadcopter. It also stores the sign of the PID values for EEPROM saving

3.2.2.6 eepromRead()

```
void eepromRead ( )
```

Read the values stored in EEPROM and store in a data structure

3.2.2.7 initializeButtonPins()

```
void initializeButtonPins ( )
```

Initializes all buttons on the remote

3.2.2.8 isInArmState()

```
void isInArmState (
            Data & d )
```

Arms the quadcopter if the gimbals are down and out

Parameters

&d	pointer to the the data structure containing throttle being sent via radio to the quadcopter

3.2.2.9 knobPressed()

```
void knobPressed (
            bool down )
```

Resets the current displayed PID or trim values

Parameters

down	the boolean which stores whether the knob is pressed

3.2.2.10 knobsUpdate()

```
void knobsUpdate ( )
```

Updates the stored knob value so that it agrees with the stored PID value displayed on the LCD screen

3.2.2.11 loop()

```
void loop ( )
```

Loop endlessly. Read gimbal positions, determine if quadcopter should be armed, determine which LCD tuning state the remote should be in, determine whether the user is inputing commands, and send data structure to quadcopter over radio.

3.2.2.12 PIDTuningState()

```
void PIDTuningState ( )
```

This function serves as a state machine. It is called when the remote is in the PID editing state

3.2.2.13 readAndMapGimbals()

```
void readAndMapGimbals ( )
```

Reads the analog gimbal positions and map to a value which may be sent over radio

3.2.2.14 readPIDArrayFromEeprom()

```
void readPIDArrayFromEeprom (
            int currentEEPROMAddressNumber )
```

Reads the PID values stored in EEPROM

Parameters

currentEEPROMAdressNumber	necessary to read correct data

3.2.2.15 readTrimFromEeprom()

```
void readTrimFromEeprom (
            int EepromAddress )
```

Reads the trim values stored in EEPROM

Parameters

EepromAddress	eeprom address to read correct data (should be at 43)

3.2.2.16 resetCoefficientValuesAndSigns()

```
void resetCoefficientValuesAndSigns ( )
```

Resets all PID coefficients.

3.2.2.17 setup()

```
void setup ( )
```

Setup the remote firmware by initializing radio, gimbal pins, LCD screen, knobs and buttons, and reading PID and trim values from EEPROM.

3.2.2.18 TrimTuningState()

```
void TrimTuningState ( )
```

This function serves as a state machine. It is called when the remote is in the Trim editing state

3.2.2.19 updateDisplayPIDElementChange()

```
void updateDisplayPIDElementChange ( )
```

Updates the entire LCD screen to the PID state. It should be called when an element changes on the LCD screen. For example: arming the controller or changing which pitch, roll, or yaw PID value is being tuned.

3.2.2.20 updateDisplayPIDNumChange()

```
void updateDisplayPIDNumChange ( )
```

Updates the numbers being displayed currently on the LCD screen. It should be called when a number changes.

3.2.2.21 updateDisplayTrimElementChange()

```
void updateDisplayTrimElementChange ( )
```

Updates the entire LCD screen to the tuning state. It should be called when an element changes on the LCD screen.

3.2.2.22 writeGimbalsToEeprom()

```
void writeGimbalsToEeprom ( )
```

Saves the calibrated gimbal values to the microcontroller's EEPROM

3.2.2.23 writePIDToEeprom()

```
void writePIDToEeprom ( )
```

Saves the PID and trim coefficients to the microcontroller's EEPROM

3.2.3 Variable Documentation

3.2.3.1 armToken

```
bool armToken = true
```

State variable for armed mode.

3.2.3.2 changeLeft

```
bool changeLeft = 1
```

Keeps track of whether we are tuning whole integer or decimal PID value.

3.2.3.3 col

```
int col = 0
```

LCD screen column to write to.

3.2.3.4 colPosStart

```
int colPosStart[3]
```

What column we are writing to LCD screen for PID tuning.

3.2.3.5 cpa

```
int cpa[3][6]
```

Coefficients Properties Array for displaying and storing PID coefficients for pitch, yaw, roll. 3 rows, 6 columns.

3.2.3.6 gimbalRawValues

```
int gimbalRawValues[4]
```

Values read by gimbal. Nominally 0 - 1024 but constrained by the potentiometer.

3.2.3.7 isInTrimMode

```
bool isInTrimMode = false
```

State variable for trim mode.

3.2.3.8 knob_btn

```
bool knob_btn = 0
```

Whether the knob button is pressed.

3.2.3.9 knob_token

```
bool knob_token = false
```

trim or PID tuning variable

3.2.3.10 largestGimbalValueReadSoFar

```
int largestGimbalValueReadSoFar[4] = {303, 318, 320, 306}
```

highest value possible by analog stick gimbal

3.2.3.11 lowestGimbalValueReadSoFar

```
int lowestGimbalValueReadSoFar[4] = { 0, 0, 0, 0}
```

lowest value possible by analog stick gimbal

3.2.3.12 PIDpos

```
int PIDpos = 0
```

PID value being tuned currently.

3.2.3.13 remoteToQuadData

```
struct Data remoteToQuadData
```

Structure for radio data. Struct "Data" initialized in radio.h.

3.2.3.14 row

```
int row = 0
```

LCD screen row to write to.

3.2.3.15 saveLargestGimbalValuesSoFar

```
int saveLargestGimbalValuesSoFar[12]
```

255 + 255 + 255 = 765 is the highest save value with EEPROM for gimbal

3.2.3.16 sign

```
int sign[9] = {1,1,1, 1,1,1, 1,1,1}
```

Sign of PID tuning coefficients for storing via EEPROM.

3.2.3.17 timesOverflowed

```
int timesOverflowed = 0
```

Keeps track of times an integer has overflowed for EEPROM writing.

3.2.3.18 top

```
String top = ""
```

Initialize string to print on top row of LCD screen for PID tuning.

3.2.3.19 trimAngle

```
float trimAngle[2]
```

Trim pitch and roll angles.

Index