Angular Rate Control for VTOL Multirotor Aircraft

Low Cost Embedded Solution for Basic Rate Control of Micro UAV Rotor Craft

Akshat Deshpande
Industrial Robotics Center
Rashtrasant Tukadoji Maharaj Nagpur University's
Oberoi Center for Excellence
Nagpur, India.

Dr. Shyamkant Limaye
Center Head of Industrial Robotics
Rashtrasant Tukadoji Maharaj Nagpur University's
Oberoi Center for Excellence
Nagpur, India.

***Abstract*—This project aims to implement rate control of multiple rotor VTOL aircraft or multirotors using a 3-axis MEMS angular rate sensor. There are a number of open and closed source flight control platforms available which focus on the very high level functions of micro unmanned air vehicles (MAV's), these functions include autonomous GPS aided/denied navigation, position control, velocity control, altitude control, waypoint navigation, etc. While these modes augment the feature set with abilities which allow an operator with very less flying skills to be able to command and control the MAV but when it comes to the recent mini drone racing revolution they are not as well suited to the needs of people flying acrobatics and racing. Drone racers require a very basic setup having the ability to provide high response times. This is essential for achieving high-G manoeuvers. The challenge is being able to synchronize the process of sending motor commands with the input signals and the data from the IMU with minimal latency. This paper aims to provide a simple cheap solution for such applications.**

Keywords—rotations; drones; gyroscopes; rate control; flying; IMU; multirotor; radio control; embedded systems.

I. INTRODUCTION

Multiple rotors VTOL aircraft or Multirotor aircraft come in variety of configurations, but so far the most popular and easy to build is the quadcopter. A quadcopter has four motors arranged in the form of a symmetrical quadrilateral; it can be a square, rectangle or a kite. The main control for aircraft is along the roll, pitch and yaw axis for stabilization. The cumulative thrust vector and gravity vector add up to give us the direction of translation. In case of rotor craft the thrust and Lift vector are generally collinear but pointing in opposite directions thus giving them the unique ability to hover. Multirotors are popular because of their simplicity and lack of mechanical complexity which comes with the conventional helicopter designs, but what is lost in mechanical complexity is made up for in electronic and software code complexity.

The reason for such code complexity is that any multicopter is naturally unstable and given the practical world constraints it will never be able to fly unless there is some sort of feedback control which periodically updates the motor/actuator speeds to stabilize the system. The various levels of control hierarchically speaking are

1. Angular Rate control
2. Angle Control
3. Altitude Control
4. Heading Control
5. Position, Velocity and Acceleration Control
6. Autonomous Navigation

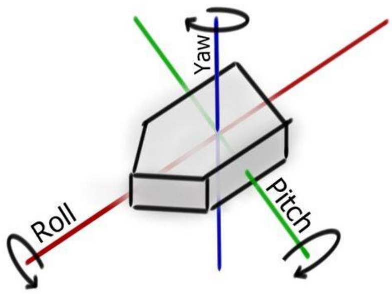

Fig. 1. Axes of rotation [1]

7. Trajectory Planning
8. Obstacle avoidance, etc.

The recent mini drone racing phenomenon has necessitated a renewed focus on the most basic level of control i.e. angular rate control, this because mini drones owing to their small size and low moment of inertia are very agile. The dynamics of this system make it very difficult to implement higher control mechanisms reliably; this is mainly because the sensors used to estimate acceleration, heading, position, etc. Simply fail to keep up with the fast attitude changes and high centrifugal forces. This makes the job of sensor data filtering much more critical. Because of this rate controlled flight has become more popular as it provides a simplistic mode of control for the pilot to showcase his/her flying abilities.

II. BASICS OF MULTIROTOR FLIGHT

Multirotors have multiple rotating propellers which generate lift in the upward direction. If this lift is more that the force of gravity exerted on the aircraft; then the aircraft should achieve flight. In order to maintain stable flight a multirotor requires active computerized stabilization where the flight controller works continuously to stabilize the aircraft on its roll, pitch and yaw axes. Fig. 1 shows the three axes of rotation and fig. 2 illustrates a quadcopter whose nose is pointing in the direction of the positive roll axis. The clockwise and counter clockwise arrows show the direction of spin of propellers. It is important to note the spin directions as they are essential for stabilization of the multirotor along its yaw axis. Fig. 3 illustrates the direction of torques generated by each motor. In this figure every spinning propeller exerts a torque on the chassis; this torque is generated as a result of the drag experienced by propellers when moving through air. As a consequence of newton's third law of motion, the propellers spinning in the clockwise direction exert a counter torque on

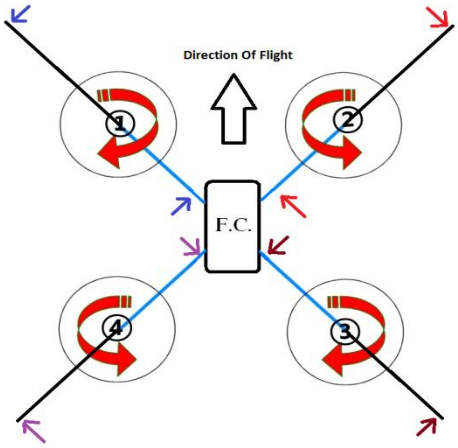

Fig. 3. Resultant torques on the frame

the copter trying to spin it in the counter clockwise direction ,likewise the propellers spinning in the counter clockwise direction exert a force on the chassis to spin it clockwise. Hence theoretically if all the motors spin at the same speed the resultant of all torques exerted on the body should approach zero and prevent the copter from spinning on its yaw axis. If the copter is commanded to yaw clockwise or anticlockwise by the pilot, then it achieves this by speeding up the corresponding diagonal motor pair and decreasing the speed on the other two motors thus creating an imbalance of torques and spinning the copter with the desired velocity. For example if the copter is commanded to yaw in the counter clockwise direction, speeds of motors 1 and 3 are increased and simultaneously speeds of motors 2 and 4 are decreased by the same amount. This way the resultant torque on the copter spins it in the desired counter clockwise direction. This is illustrated in fig. 4. The amount of increase or decrease in motor speeds influences the amount of reaction torque exerted on the body and hence the speed of rotation or angular velocity along the yaw axis [2]. The underlying assumption is that the center of gravity lies at the intersection of the roll, pitch and yaw axes. Further if we see how the copter tries to roll under pilot command. In fig. 5 the motors on the left side (Red) of the roll axis are sped up while the motors on the right side (Green) are slowed down this causes an increased coefficient of lift on the left side as compared to the right side of the copter, hence rolling the copter towards right. If we want to roll the copter left it is vice versa. The amount of increase or decreases in motor speeds directly controls the difference in lift generated on either side of the copter and thereby the speed of rotation or angular rate along the roll axis

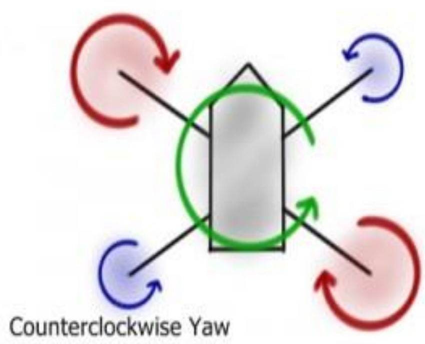

Fig. 4. Yaw control [1]

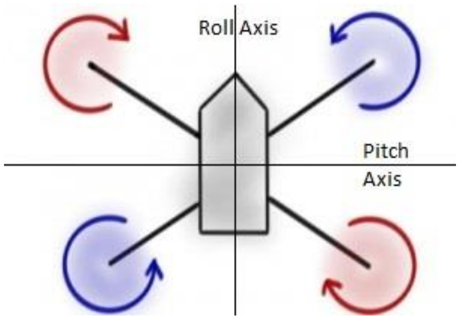

Fig. 2. Quadcopter Roll and Pitch axes [1]

[2]. A similar method is used to make the copter pitch up and down depending on controls applied by the pilot. As shown in fig. 6 when the motors on the rear (Red) of the copter are sped up and the motors on the front of the copter (Green) are slowed down, there is more lift generated on the rear half of the copter as compared to front half, this difference in lift exerts a force on the copter causing it to pitch forward. If the copter has to be pitched backwards we just switch the motors being sped up with the ones being slowed down. The amount of increase or decrease in motor speeds decides the difference in lift generated on either side of the copter and hence the speed of rotation or angular rate along the pitch axis [2].

$$\textit{Considering Figure } 3, \textit{if we assume}$$

$$\textit{Total Thrust } F = f1 + f2 + f3 + f4 \quad (1)$$
$$f1 = \textit{Thrust generated By Motor } 1$$
$$f2 = \textit{Thrust generated By Motor } 2$$
$$f3 = \textit{Thrust generated by Motor } 4$$
$$f4 = \textit{Thrust generated by Motor } 5$$

$$\textit{Considering the propellers used are of the same size and pitch}$$
$$\textit{same size and pitch}$$

$$\textit{Thrust } \propto \textit{Motor Speed} \quad (2)$$

$$\textit{Considering all motors are generating}$$
$$\textit{thrust of } 500 \textit{ gms each}$$

$$f1 = f2 = f3 = f4 = 500 \textit{ gms}$$
$$\textit{Total thrust} = 2000 \textit{ gms}$$

$$\textit{To yaw the copter clockwise,}$$
$$\textit{following will be the corresponding,}$$
$$\textit{changes in motor thrusts}$$

$$f1 = 300 \; f2 = 700 \; f3 = 300 \; f4 = 700$$
$$\textit{Total thrust} = 2000 \textit{ gms}$$

$$\textit{Similarly for rolling to the right,}$$
$$\textit{following will be the corrsponding,}$$
$$\textit{changes in motor thrusts}$$

$$f1 = 700 \; f2 = 300 \; f3 = 300 \; f4 = 700$$
$$\textit{Total thrust} = 2000 \textit{ gms}$$

$$\textit{Finally for pitching the aircraft}$$
$$\textit{forward the corresponding motor}$$
$$\textit{thrusts will be as follows}$$

$$f1 = 300 \; f2 = 300 \; f3 = 700 \; f4 = 700$$
$$\textit{Total thrust} = 2000 \textit{ gms}$$

Here the thrust increase/decrease of 200gms is completely arbitrary, in the practical implementation this value will be

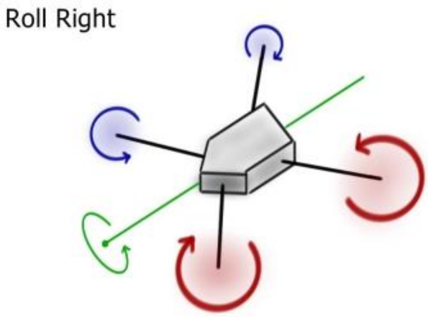

Fig. 5 Roll Control [1]

influenced by the desired angular rate and output of the control loop, nevertheless these equations tell us that for any manoeuver even though the individual motor thrusts change corresponding to the axis of motion but the total thrust generated by the system remains same. This is to ensure that the copter always generates the same lift for a certain throttle level and the copter does not descend or ascend due to application of control input. Though after the copter tilts the magnitude of thrust in the downward direction reduces causing the copter to descend, but this scheme is important to have symmetric control over all axes of rotation [2].

Here 500 grams is the base thrust and the motor speed signal corresponding to the base speed is called the collective. The value of this collective is specified by the pilot's throttle signal. Also complimentary control helps when the motors are saturated. For example when the motor are operating close to 90% of their achievable thrust and the pilot demands a roll rate of 400 degrees per second at that point some motors will be required to spin at 110% while others will be required to spin at 70% of their achievable speeds but since the motors cannot spin at more than 100% speed the response of the system suffers, even then the system is still maneuverable considering that the rest of the motors can slow down to 70 % speed and produce a rotation rate close to the desired rate. To work around this problem the maximum collective applied to the motors is limited to 85% of the total motor output signal, this way there is enough headroom for accommodating rate requests of close to 300 degrees per seconds without compromising on system response [3].

This is even more useful when the collective is close to zero i.e. if the collective is at 30% and the controller demands a rate of 360 degrees per seconds on the pitch axis, in this case two of the motors will have to spin at 0% speed while the other two will have to spin at 60% speed. This might be fine to do theoretically but when a brushless motor stops, it can be difficult to restart it, this is because the way BLDC motors are driven they do not produce a lot of starting torque and with air flowing through the propellers the motor might fail to restart in air and lead to a crash. Hence the software should ensure that it never lets the motor speeds fall below a certain minimum value to protect against motor stalling. This minimum value is close to 15% of the throttle signal. In this case as well because the motor speeds cannot be decreased beyond a certain value, the complimentary thrust mechanism helps the flight controller to comply with the pilot's desired commands while flying at very low throttle levels, all of this without compromising on flight integrity [3].

A. *Abbreviations and Acronyms*

IMU inertial measurement unit; VTOL vertical take-off and landing; MAV micro aerial vehicle; UAV unmanned Aerial vehicle; MEMS micro electro-mechanical sensors; GPS global positioning system; GNSS global navigation satellite subsystem; BLDC brushless direct current.

III. IMPLEMENTATION

For the purpose of this implementation an AVR microcontroller will be used; however this scheme can be diversified so that it can be implemented using microcontrollers with more powerful architectures. Specifications of the test system are as follows.

A. AVR Atmega328 16Mhz 5V TTL on an Arduino Nano.
B. Invensense ITG-3200 3-Axis MEMS Gyro connected over the I2C bus

IV. CHARACTERISTICS OF INPUTS AND OUTPUTS

Inputs to the system include a MEMS gyroscope connected on the I2C bus and a radio control receiver connected to the port pins which support pin change interrupts. The I2C bus is configured to communicate at 400 KHz, it is externally pulled up to 3.3 volts so that both the micro-controller as well as the sensor can communicate without damaging each other. The receiver has four wires each corresponding to four channels of roll, pitch, yaw and collective control connected to pins PB0-PB3 of the controller these four signals communicate at 50Hz on a protocol based on PWM which is shown in figure 7, decoding this signal has been described in the focuses ahead. Output of the system is in the form of four 400Hz PWM signals, one for each motor of the quadcopter. This signal communicates the speed of motors as commanded by the flight controller to the electronic speed controllers which spin the motors at this desired target speed. The number of signals depends on the number of motors for the configuration of multicopter being controlled. This signal protocol is identical to the one used by the radio receiver to communicate position of sticks on the pilot's radio transmitter module to the flight controller, but this operates at a much higher frequency of 400 Hz to provide a higher refresh rate of motor speeds [3]. The speed controllers are connected to pins PD0-PD5. This controller is able to control multicopter configurations having up to 6 motors.

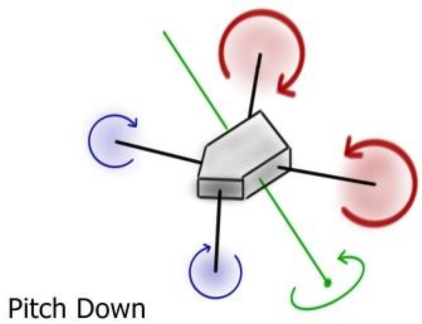

Fig. 6. Pitch control [1]

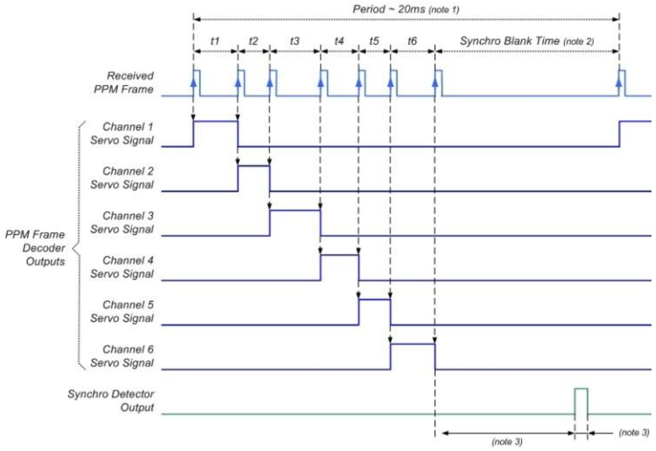

Fig. 7 PPM and PWM signal waveform [4]

V. THE RC COMMUNICATION PROTOCOL

While the MEMS gyroscope communicates through the standard Phillips I2C protocol, the radio receiver communicates using a protocol which is called PPM or pulse position modulation in the RC hobby community; in reality this is in fact a PWM signal where the width of pulse gives us the value of signal. This signal is illustrated in the fig. 7. The figure shows that each individual PWM signal corresponds to one channel of control, with regard to this implementation there are only four channels and hence four PWM signals to decode. Each PWM signal operates at a base frequency of 50 Hz and hence the period of this signal is 20 milliseconds. Out of this entire time period of 20 milliseconds the signal is always ON for the first 1000 microseconds and changes in width only happen between 1~2 milliseconds, while for the remaining 18 milliseconds the signal is always off. During this time the signal does not carry any information. When width of the pulse is 1 millisecond or 1000 microseconds this means that the value of signal it represents is 0 % and when width of the pulse is 2 milliseconds or 2000 microseconds the value of signal is 100%. Different radio manufacturers use the same protocol but the pulse widths corresponding to 0% and 100% signal might be different. For example the radio used here has a minimum pulse width of 828 microseconds and a maximum pulse width of 2172 microseconds. This necessitates that whenever a new brand of receiver is connected to the system, the user should first calibrate the radio to ascertain the maximum and minimum pulse widths. These values are saved in the EEPROM of the AVR controller and it is not required to repeat this calibration procedure every time the system is powered up [3].

The output pins of the flight controller communicate desired motor speeds to the speed controllers using this same protocol but in their case the unusable pulse width of 18 milliseconds is reduced to only 0.5 milliseconds this way the maximum period of this pulse is reduced to 2.5 milliseconds instead of 20 milliseconds thereby increasing the frequency to 400 Hz instead of 50 Hz. One important consideration when choosing the electronic speed controllers is that they should support a 400 Hz refresh rate. The distinction between speed controllers which support fast update and those who don't is very essential to understand because the effects on flight performance are significant [3].

Output from the flight controller has a maximum pulse width of 2000 microseconds corresponding to 100% signal and a minimum width of 1000 microseconds corresponding to 0% signal. The speed controllers are unaware of the pulse widths corresponding to 0% and 100% signal generated by the flight controller. To make speed controllers compatible with different pulse widths generated by different manufacturers of radio control equipment they have a calibration routine built in to them which lets the user program the maximum and minimum pulse widths generated by any receiver or flight controller. The user has to invoke this subroutine to let the speed controllers know the maximum and minimum pulse widths to be expected from the flight controller.

VI. INITIALIZATION

The code begins by initializing the gyroscope. It sets the range of the gyro at +- 2000 degrees per second on each axis, the internal sample rate is set at 8K samples per second and the internal LPF is set to band limit the gyro data to below 256 Hz. The first 50 filtered gyro readings on each axis are fed to a moving average filter. The output is checked to see if resultant angular rate is not more than 30 degrees per second on each axis, if this is not the case then either the aircraft is moving or the sensor is unhealthy.

Timer modules within the AVR controller are used to measure the pulse widths of signals coming from the radio receiver. They are also used for calculating delays and for generating required pulses to instruct the speed controllers to spin motors at their designated speeds.

Timer0 is programmed with a prescaler of 8 this ensures that the time elapsed per tick is 0.5 microseconds. This way time required for the timer to count 256 steps and overflow is 128 microseconds, an overflow counter is maintained which keeps track of number of overflows since start-up. To find the current time since power up the number of overflows is multiplied with 128, then the current value of timer0 is divided by 2 and added to this product. This gives us the exact time elapsed in microseconds. This timer is used for calculating delays and also for measuring pulse widths for the receiver signals.

Timer1 is programmed in fast PWM mode with a prescaler of 8 to ensure one timer tick happens after 0.5 microseconds,

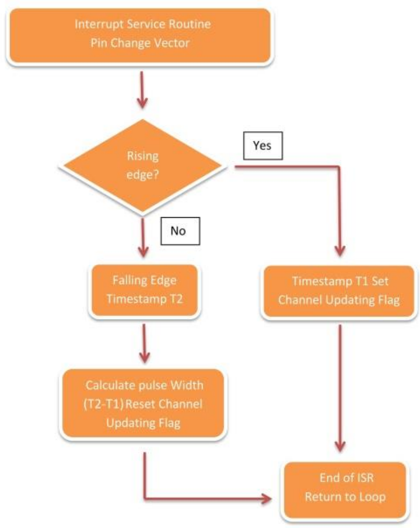

Fig. 8 PPM signal decoder

frequency of PWM is decided by the top of counter which is specified by the value of the input capture register. The value of ICR1 is set to 3999 to generate a wave with frequency of 400Hz. Values within the output compare match registers and the output compare interrupt handler is used to generate up to 6 simultaneous PWM signals for sending motor commands to the motor speed controllers.

Input channel pulse widths from the radio control receiver are sanity-checked for errors, if this is not checked then it might lead to uncertain values being fed into the rate controller, if the sanity checker produces an error then either the receiver is disconnected or some inputs are disconnected. In both cases the initialization will fail and the controller will generate a specific beep sequence to warn the user of an initialization error. Prerequisites for initialization are that the platform is set motionless and receiver is connected on power up.

VII. START-UP

During the start-up sequence the gyroscope is calibrated. As a part of this sequence the gyro data is averaged over 3 seconds and the gyro bias per axis is calculated, these values are used during flight for estimating the real gyro angular rate sans error due to gyro drift. Temperature readings from the gyro sensor are used to make small adjustments to the estimated gyro bias. Once the gyroscope is calibrated, the platform is ready to allow its control loop to take over.

VIII. ARMING

As a safety feature arming is a sequence which is enforced upon by the flight controller, this way the pilot has to explicitly specify when the motor outputs are enabled thus allowing actuation of the propellers.

IX. INPUT CONVERSION TO THEIR PHYSICAL REPRESENTATIONS

The data from the radio receiver is decoded by using a pin change interrupt method. As shown in fig. 7 the receiver channel pulse widths appear in a staggered fashion such that no two channels have pulses occurring at the same time. The pulses appear one after the other and therefore the signals are decoded in the order of their occurrence. When all four channels have been decoded, it is implied that a new data

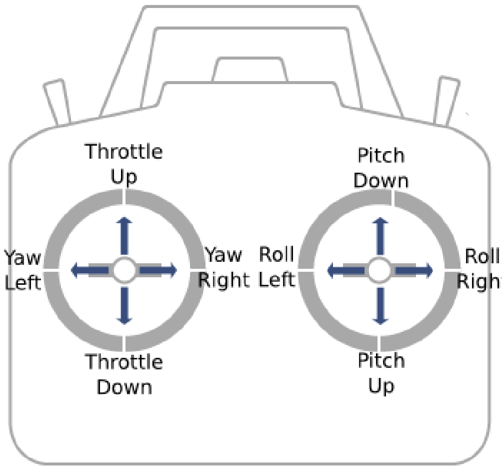

Fig. 9 RC transmitter stick configuration [5]

frame has arrived. Fig. 8 shows the how receiver channel widths are measured by the software. The receiver pulse widths ideally range from 1000~2000 microseconds but the practical pulse widths in this test case range from 828~2172 microseconds, from this measured value, the value of pulse width which corresponds to 0% signal value is subtracted. In the ideal case 0% signal is represented by 1000 microseconds so the resultant pulse width range after subtraction is reduced to 0~1000 microseconds. In the test case the 0% signal corresponds to a pulse width of 828 microseconds so after subtraction the range is 0~1344 microseconds. The code constrains values of the pulse widths from 0~1344 so that values falling out of this range are limited to a maximum of 1344 and a minimum of 0. These constrained values are subtracted from their respective pre-constrained counterparts. The difference between the two should be zero at all times, if the difference is greater or less than zero then this indicates that invalid signals are being received from the receiver and can be ignored. Each time an invalid signal is detected the code increments an error counter variable, if the value if this variable reaches 100, this means that the signal has been invalid for 2 seconds and there is some problem with the RC communication link. In which case the all the motors are shutdown as a safety feature. This error variable is reset to zero every 10 seconds, just so that small receiver errors don't accumulate and reset the system.

Now the code has data corresponding to the stick positions on the pilot's radio transmitter module. Fig. 9 shows a typical layout of the pilot's transmitter module, it has two sticks which can move in the up/down and right/left directions. The pulse widths are converted to their corresponding stick positions. This is done by mapping the signal pulse widths. For the throttle stick, the pulse width range of 0~1344 is mapped to 0~1000, since it is not spring loaded and does not return to the center when released. Similarly this is repeated for all the other channels but since all these sticks are spring loaded the pulse widths of 0~1344 are mapped from -500~500 to denote positive as well as negative stick deflections [5]. By using this method the position of any stick can be estimated within 1/1000 of its range. The gyroscope data which is acquired over the I2C bus is in the form of three 16-bit integers, one each for X, Y and Z axes is converted to angular rate in degrees per second by multiplying it with the sensitivity scale factor of 14.375. After this the software has raw angular rate data in degrees per second for the roll, pitch and yaw axes. Now there are two decoded data inputs to the system in the form of stick position and gyro rate. Before feeding these values into the control loop, the code has to specify what each individual stick position stands for. Since the code aims to implement rate control, the position of roll, pitch and yaw sticks are converted into the desired angular rate measured in millidegrees per second. The throttle stick position value specifies the amount of collective from 0~1000. The yaw stick position specifies the desired yaw angular rate in +-2000 millidegrees per second, likewise the roll and pitch stick positions specify the desired roll and pitch angular rate in +-3600 millidegrees per second. These ranges can be changed depending on the pilot's preferences of agility; making them these ranges higher will make the copter more agile while decreasing them will make it more docile. This is called stick

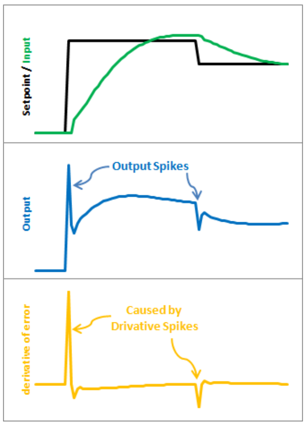

Fig. 11 Derivative kick [6]

response scaling and is achieved by mapping the stick position ranges with the desired range of angular velocities.

Mapping is a function used frequently for rescaling data ranges, this function is derived from the Arduino libraries and this is the function definition.

$$Long\ map\left(\begin{array}{c} long\ x, long\ oldmin, \\ long\ oldmax, long\ newmin, long\ newmax \end{array}\right)$$

$$x = (x - oldmin) * \frac{newmax - newmin}{oldmax - oldmin} + newmin \qquad (3)$$

Another frequently used function from the Arduino libraries is the constrain function defined below

$$\begin{array}{c} constrain\ (long\ x, long\ minvalue, long\ maxvalue) \\ if\ (x < minvalue) \rightarrow x = minvalue \\ if (x > maxvalue)\ \rightarrow\ x = maxvalue \\ return\ (x) \end{array} \qquad (4)$$

X. THE MAIN CONTROL LOOP

We use a Proportional Integral (PI) controller for controlling the angular rates along the 3 axes of motion; this requires the code to run 3 separate instances of the control algorithm, one for each axis. The outputs of all three controllers are converted to 3 motor speed signals and applied together. Fig. 10 shows how the PI controller works. The proportional gain takes care of the impulse response of the system. Using only a proportional controller for this system would be enough for stability, making it the most important parameter which decides if a system is flyable or not, but the controller might not be able to compensate for the steady state error, this is because the proportional controller generates a constant output for constant error and under external forces like winds, the constant output of the controller might not be enough to counteract these forces. To reduce this steady state error to

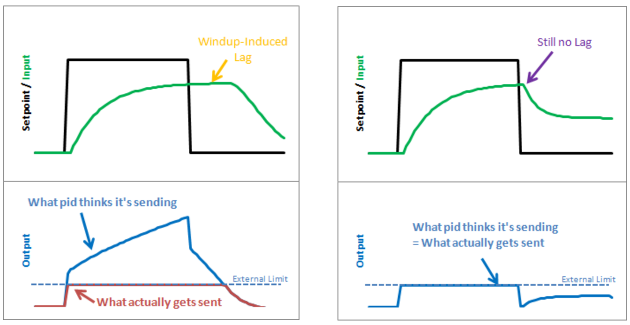

Fig. 12 Integral windup mitigation [6]

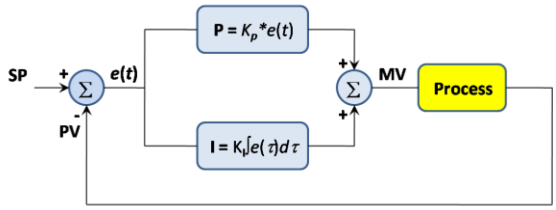

Fig. 10 PI controller

zero we use the integral gain constant which accelerates the system towards the set point and acts to reduce the steady state error. The derivative term is not used in this controller since the transient response of this system isn't very large, also introduction of a derivative term can introduce instability in the system due the characteristic derivative kick whenever a step change in set point is provided to the system, this kick can cause momentary saturation of the motors and excessive current draw. Fig. 11 shows the derivative kick phenomenon [6][7].

In a PI controller of this kind we have to protect against integral windup, this is done by constraining the I term between two values so that system does not get saturated at any point due to a longer lasting error signal [6][7], windup mitigation is illustrated in fig. 12.

One very important assumption is that the loop time of the controller is small enough that three rotations can be fused in to one; this ensures that we compensate for any rate error on all three axes at the same time, without violating the non-commutative property of 3D rotations. Further as a part of the control loop the gyro data which is in degrees per seconds is converted to millidegrees. The reason why we increase the range of input data by converting the units into smaller units is because we want to avoid computing floating point data, which can take longer for the processor. This also gives us more resolution to work with. Also the rate at which the controller completes an iteration is not affected by the speed at which the inputs and outputs of the system are generated. The process of getting the inputs and generating the outputs are completely decoupled from the main loop, this ensures that the input signal from the pilot appearing every 50Hz along with the gyro signal at 256Hz and motor outputs being generated at 400Hz do not cause unwanted control loop delays or over-runs. To make these three signal frequencies work with the main loop in stable manner we perform scheduling using timer interrupts, this ensures that multiple subroutines are serviced while having minimal impact on the rate of execution of the main code. Next the rate error is calculated by subtracting the actual rate measured by the gyro from the desired rate calculated from the pilot's input. There will be three errors

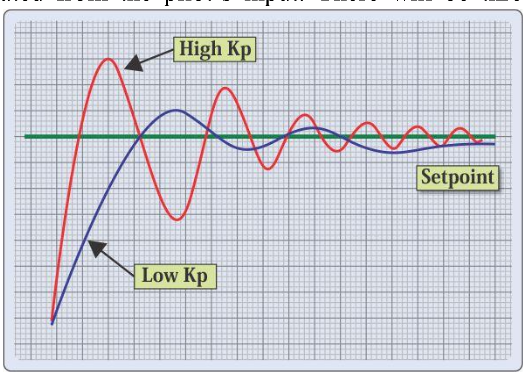

Fig. 13 Effects of proportional term on system response [8]

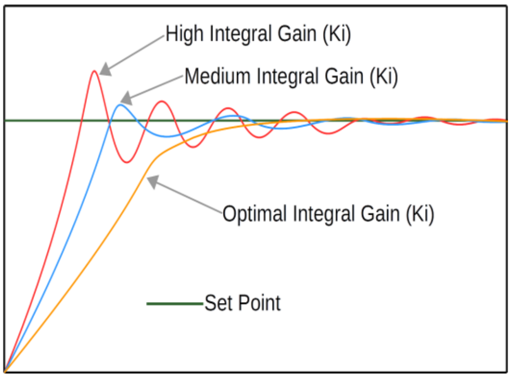

Fig. 14 Effects of integral term on system response [9]

calculated on the roll, pitch, and yaw axes. These errors will be given as inputs to the PI algorithm. The rate error is multiplied by the proportional gain this constitutes the P-term. Further the current error is added to the last error which was measured during the previous iteration, this accumulated error value is multiplied by the loop time to calculate the integration of the error signal. This integrated error value is multiplied by the I-gain constant to calculate the I-term. The addition of P and I terms forms the output of the PI controller [8].

PI Controller

$$\mathbf{c(t) = P * e(t) + I * \int_{t1}^{t2} e(t)\, dt} \quad (5)$$

P – *Proportional Gain*
I – *Integral Gain*
$t2$ – *Time at begining of present iteration*
$t1$ – *Time at end of previous iteration*
dt – *Loop Time*
$e(t)$ – *Rate Error*
$c(t)$ – *Controller Output*

Fig. 13 shows the effect of only P term on the output, fig. 14 shows the effect of I term on the output and fig. 16 shows the desired response from the combination of PI terms.
Output of the controller is converted to corresponding motor speed signals which act to reduce the error and providing accurate control. The fig. 15 shows how motor output PWM signals are generated using timer interrupts. Equations governing the motor speed changes as a result of the output of the controller are shown below; here the quadcopter shown in fig. 3 is taken as reference.

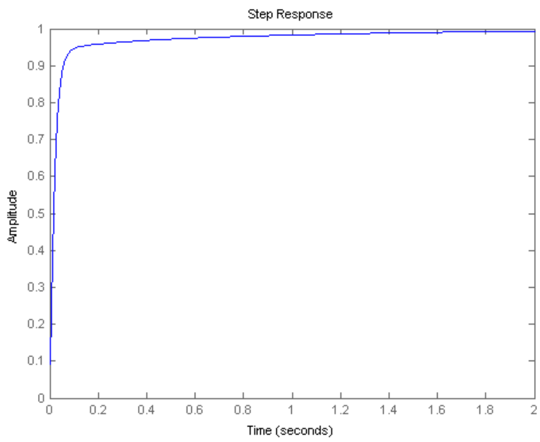

Fig. 16 Desired output [10]

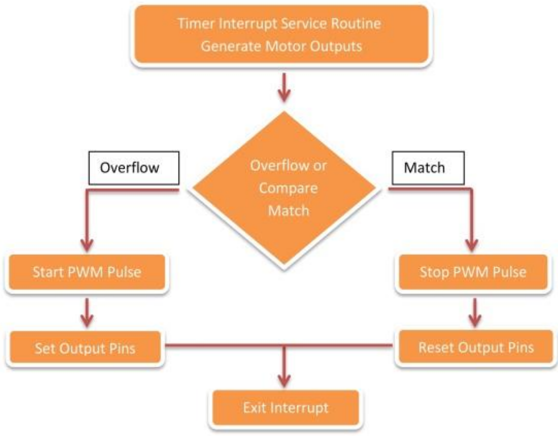

Fig. 15 Motor signal generation

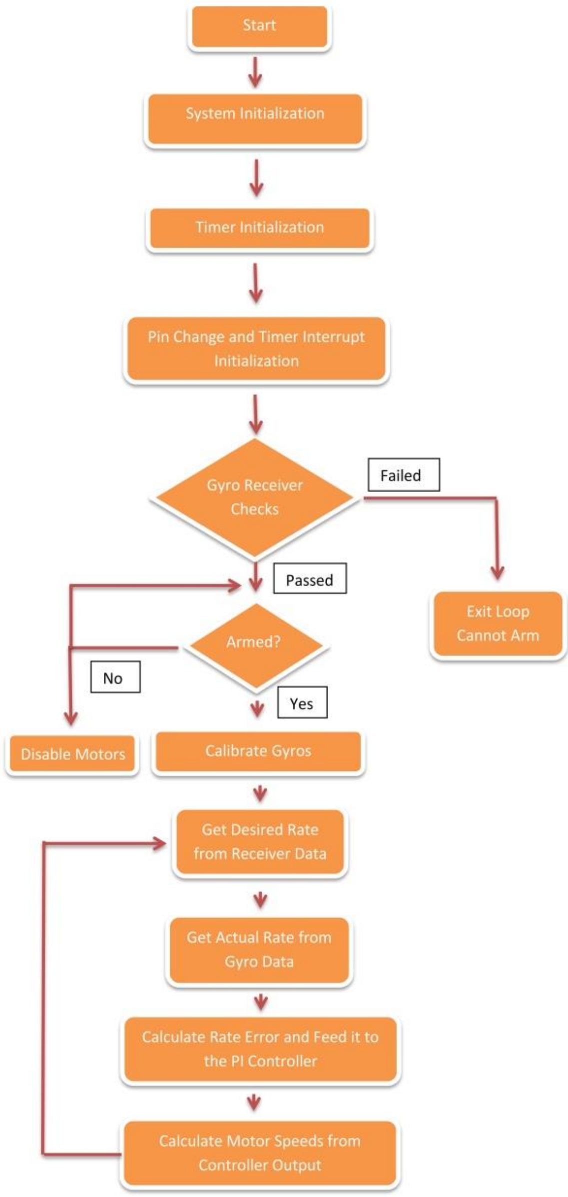

Fig. 17 Software flow diagram

$$\textit{Changes to Motor Outputs by the Yaw controller} \tag{6}$$

$$\begin{aligned}
Motor\ 1\ Output &= Present\ Motor\ 1\ Output - Yaw\ Output\ Signal \\
Motor\ 2\ Output &= Present\ Motor\ 2\ Output + Yaw\ Output\ Signal \\
Motor\ 3\ Output &= Present\ Motor\ 3\ Output + Yaw\ Output\ Signal \\
Motor\ 4\ Output &= Present\ Motor\ 4\ Output - Yaw\ Output\ Signal
\end{aligned}$$

$$\textit{Changes to Motor Ouputs by the Pitch Controller} \tag{7}$$

$$\begin{aligned}
Motor\ 1\ Output &= Present\ Motor\ 1\ Output + Pitch\ Output\ Signal \\
Motor\ 2\ Output &= Present\ Motor\ 2\ Output + Pitch\ Output\ Signal \\
Motor\ 3\ Output &= Present\ Motor\ 3\ Output - Pitch\ Output\ Signal \\
Motor\ 4\ Output &= Present\ Motor\ 4\ Output - Pitch\ Output\ Signal
\end{aligned}$$

$$\textit{Changes to Motor Ouputs by the Roll Controller} \tag{8}$$

$$\begin{aligned}
Motor\ 1\ Output &= Present\ Motor\ 1\ Output + Roll\ Output\ Signal \\
Motor\ 2\ Output &= Present\ Motor\ 2\ Output - Roll\ Output\ Signal \\
Motor\ 3\ Output &= Present\ Motor\ 3\ Output - Roll\ Output\ Signal \\
Motor\ 4\ Output &= Present\ Motor\ 4\ Output + Roll\ Output\ Signal
\end{aligned}$$

The motor outputs from the PI controller are constrained to protect against motor saturation as well as stalling. The motor speeds are sent to the motor speed controllers at the next update of the 400Hz loop. This process is repeated continuously to ensure that the desired and actual rates follow each other on the roll pitch and yaw axes up until the time that the system is not disarmed by the pilot. Tuning of the PI controller on each of the three axes is done experimentally, such that the pilot is comfortable with the response. This helps different pilots customize the feel of the aircraft in accordance with their requirements.

XI. RESULTS AND CONCLUSION

The control algorithm described in the above text was implemented using the prior mentioned test bed and the flight characteristics were tested. Implementation using an 8-bit controller helps keep the cost down at the same time it allows pilots to showcase their flying capabilities. Higher levels of control can be ported on top of this controller for use with SWARM applications, the kind being used at the flying machine arena at ETH Zurich in Switzerland. The performance of this controller has been showcased in the video below
https://www.youtube.com/watch?v=qm6R5rheX40

The controller compensates well given the imperfections in this homemade frame. This demonstrates the robustness of this controller. Fig. 17 illustrates the main control loop

REFERENCES

[1] http://blacktieaerial.com/the-physics-of-quadcopter-flight/
[2] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ control techniques applied to an indoor micro quadrotor," IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2451–2456, 2004.
[3] J. Escare˜no, C. Salazar-Cruz, and R. Lozano, "Embedded control of a four-rotor UAV," American Control Conference, vol. 4, no. 11, pp. 3936–3941, 2006.
[4] http://diydrones.com/profiles/blogs/705844:BlogPost:38393
[5] http://smaccmpilot.org/hardware/rc-controller.html
[6] http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/
[7] Jun Li, Yuntang Li (2011). ―Dynamic Analysis and PID Control for a Quadrotor‖ 2011 International Conference on Mechatronics and Automation.
[8] http://www.pcbheaven.com/wikipages/PID_Theory/
[9] http://coder-tronics.com/pid-tutorial-c-code-example-pt1/
[10] http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=ControlPID