AUTOPILOT CONTROL SYSTEM - IV

CONTROLLER

The data from the inertial measurement unit is taken into the controller for processing. The input being analog requires to be passed through an ADC before being processed. Kalman filter is used for the estimation of the states and then fuzzy controller comes into picture. The controller also interfaces with the LCD panel, the memory card and also the computer for the logging of data.

ATMEGA32 Microcontroller

The microcontroller used is ATMEGA32. This is because we have previous experience on working with ATMEGA series microcontroller. It has a 32KB flash memory for program storage, 2KB RAM, 8 ADC channels, 3 timers with PWM support (used for servo control). These functionalities are sufficient for the blocks we had to create code. Various modules of codes were implemented for the different blocks (Kalman filter, Fuzzy controller, LCD interfacing etc.).

Kalman Filter

The measured data is noisy and the process of navigation also is not precise. Kalman filter is a way to get a best estimate about the process variables (position i.e. location and orientation of our plane) from these noisy measurements. It is an efficient recursive filter that estimates the state of a linear dynamic system from the series of noisy measurements. A recursive estimator means that only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state.

The following basics equations (4 - 8) were implemented in the code:

$$\bar{\mathbf{x}} = \Phi \hat{\mathbf{x}} \quad \textbf{(Equation 1)}$$

Where Ø the state transition matrix and x is the state variable

$$\bar{P} = \Phi P \Phi^T + Q \quad \text{(Equation 2)}$$

Where P is the covariance matrix

$$\hat{x} = \bar{x} + K(y - M\bar{x}) \quad \text{(Equation 3)}$$

Where K is the Kalman filter gain

$$K = \bar{P} M^T (M\bar{P}M^T + R)^{-1} \quad \text{(Equation 4)}$$

Where M is the measurement matrix

$$P = (I - KM)\bar{P}(I - KM)^T + KRK^T \quad \text{(Equation 5)}$$

Fuzzy Controller

A fuzzy control system is a control system based on fuzzy logic - a mathematical system that analyzes analog input values in terms of logical variables that take on continuous values between 0 and 1, in contrast to classical or digital logic, which operates on discrete values of either 0 and 1 (true and false).

Since in autopilot we require the control system to behave similar to a human response we planned to use fuzzy logic. It has the advantage that the solution to the problem can be cast in terms that human operators can understand, so that their experience can be used in the design of the controller. This makes it easier to mechanize tasks that are already successfully performed by humans.

The following is an overview of the fuzzy control system. They consist of an input stage, a processing stage, and an output stage. The input stage maps sensor or other inputs, such as switches, thumbwheels, and so on, to the appropriate membership functions and truth values. The processing stage invokes each appropriate rule and generates a result for each, then combines the results of the rules. Finally, the output stage converts the combined result back into a specific control output value. These stages are respectively referred to as fuzzification, fuzzy inference with knowledge base and defuzzification. Fig 4.2 shows these stages as block diagrams.

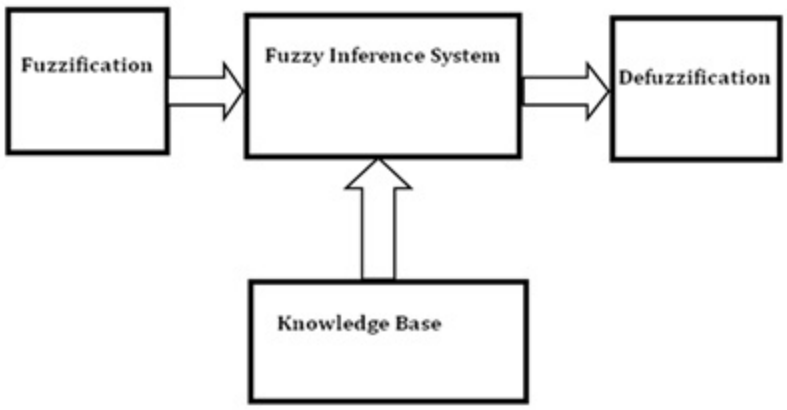

Fig 4.2 - Fuzzy Controller in blocks

We implemented the basic requirements for a fuzzy controller in assembly code for ATMEGA32. Triangular transfer function was used as membership functions. Codes were written and implemented in the microcontroller.

Output Interfacing Units

The system has to be interfaced with the peripherals. Code is written in the microcontroller for these interfacing modules. The modules include LCD interface, memory card interface, and computer interface. Servo control also needs to be done with the microcontroller.

LCD Interface

This is to have an output device to print out variables values and messages during test runs of the code. This interface would also be used to display messages from the autopilot later in the design. It was tested on hardware and is working satisfactorily.

Memory Card Interface

While programming for the accelerometers we came to the conclusion that we need to have to save the data from the accelerometer over a run to later work with it and study both its output (and thereby calibrate) and also working of our controller by saving its parameter and variable values during the running of the code. We needed a storage device to store large amount of data.

Of the two alternatives for a light weight but sufficient memory capacity (USB pen drives and MMC/SD card used in mobile phones) we decided to use MMC cards since they are lighter and easy to interface. This interface was created and tested on hardware. MMC cards use an SPI interface with some commands for transfer of data. We implement FAT32 file system so that we can read our data on any operating system (We are now using windows XP). We can now dump any amount of data (up to 4 GB based on capacity of card) for later analysis.

Computer Interface

Interfacing with the computer enables the data logging of accelerometer and gyroscope which is important for calibration purposes. To calculate the offset errors in the accelerometer and gyroscopes, we take a set of sample values and the error covariance is calculated in the

computer. This shows how much reliable the accelerometer and gyroscope readings could be and thus we can set the weightage for both sets of values. USB interface is the mode of transfer of data between the microcontroller and the computer .The code has been implemented for the USB protocol.

Servo Control

This is implemented to control the servo motors to affect the change in path specified by the processors (fuzzy controller) to maintain the desired orientation. This basically involves enabling the timers and generating appropriate PWM signals for the required angle.

Since the microcontroller doesn't have required number of timers appropriate for servo control we have implemented it in software. We get an accuracy of about .1 degree resolution for each servo.

ACTUATORS

The change in path has to be brought into action using a set of motors and we are using a set of servo motors for that purpose. The processing unit outputs the change in path required to maintain the desired path. This is communicated to the servos through the servo control unit.

Servo Motors

The motors we used for testing were Futaba S3003 model servos.

Fig 5.1 Servo motor

Conclusion

An autopilot control system was successfully designed, implemented and developed. The inertial measurement unit comprising of the accelerometer and gyroscope were calibrated and mounted on the glider. These devices measure the acceleration and tilt of the glider plane and log the data onto the onboard memory card. The deviation from the desired path and orientation, calculated by the central processor, was corrected by the servo motor set which maintained the tilts of the plane at desired levels. In-flight data is also available for analysis since it was logged onto the memory card.

Future scope of work

Processor limitations restricted the Kalman filter implementation to one-dimensional only. Because of the same limitations, in the processing stage of the fuzzy controller, the rules were based on one dimensional values only (either pitch or roll or theta and not a combination of 2 or more of these angles).

These limitations can be removed by upgrading the controller and the software part. The basic hardware can remain the same. So we have the essential setup to upgrade the system to a fully fledged autopilot in the future.

Source : http://www.botskool.com/tutorials/fourth-year-projects/autopilot-control-system/autopilot-control-system-page-4