ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

DEVELOPMENT OF DATA LOGGER FOR MAV USING FREE RTOS ON PIC32

Pankaj Akula*, Yamuna V, C.M Ananda, Mr. Niranjana Swamy G S
* Scientist Aerospace Electronics and Systems Division, CSIR National Aerospace Laboratories Bangalore India.
Project Intern Aerospace Electronics and Systems Division Kalpataru Institute Of Technology Tiptur India.
Senior Principal Scientist Aerospace Electronics and Systems Division, CSIR National Aerospace Laboratories Bangalore India.
Assistant Professor Aerospace Electronics and Systems Division Kalpataru Institute Of Technology Tiptur India.

ABSTRACT

In this paper integration of Micro Electro-Mechanical Systems (MEMS) based Inertial Measurement Unit (IMU), Magnetometer, Global Positioning System (GPS) and Pressure sensors for data logging using PIC32 is proposed. The flight data parameters such as acceleration, altitude, pressure, location and temperature, pitch, roll, yaw are proposed to be recorded on the data logger. PIC32 micro-controller is used as a main control unit for sensor integration with MPLAB X IDE. Interface of all sensor with controller using FreeRTOS gives multitasking capabilities. FreeRTOS is a real time operating system that is used for implementing the application layer. The read time for different sensors and execution time of the Nonlinear Complementary Filter (NCF) estimator in Data logger are recorded. The data log time using Free RTOS for Barometric pressure (BMP) sensor is in the order of 7.96ms, for IMU in of the order of 0.49ms, for Magnetometer in of the order of 0.26ms and for GPS in the order of 0.18ms.

KEYWORDS: PIC32-microcontroller MAV; MEMS sensors; Free RTOS; IMU;.

INTRODUCTION

Data logger is responsible for writing data on to flash and is dedicated to collect temperature, pressure, speed, altitude, acceleration and angular rates. Micro Aerial Vehicle (MAV) are designed to be very small and their performance greatly depends on onboard sensors. The MAV has a variety of potential uses in military applications such as surveillance, local reconnaissance, and control of fire and intruders detection [1]. The onboard sensors will collect data from the external environment, process them and will give the output in the digital form. The Accelerometer, Pressure, Gyroscope, Magnetometer and GPS sensors are the critical sensors which will be used in MAV for navigation [2]. These sensors will periodically receive the pressure, temperature, altitude, pitch, roll, yaw in x, y, z co-ordinates, latitude and longitude, speed, date and time. In this project data logger logs data from onboard sensors on the MAV and it will be sent to the ground station for monitoring. In order to make the MAV autonomous it is equipped with Autopilot, Camera and On-board Sensors. MAVs are based on MEMS based systems. Typical sensors used in MAV are BMP sensor for pressure and temperature, IMU for attitude estimation, GPS and so on. The system also interfaces with the telemetry. The Real Time Operating System(RTOS) provides the time constraints for the tasks to complete its operations based on the priorities assigned [3].The time constraints are given to the sensors to perform its operation of obtaining the real time data. "A conventional attitude determination system, which is made up of MEMS inertial sensors for Micro Inertial Measurement Unit (MIMU) based on the strap-down attitude determination is hard to satisfy the long time high accuracy because of the measurement errors increasing with time due to the random drift of inertial sensors" [4] . Clark N.Taylor [5], explains about the sensors such as IMU, GPS which are used for navigation of MAV. Syed Kamal [6], describes the real time kernel which provide multitasking capabilities and round robin scheduling algorithm based on PIC16 controllers where each task has been prioritized using FreeRTOS where low priority tasks are pre-empted by the high priority tasks, Then low priority tasks goes into running state and then high priority task will be in the Ready/Idle state, if the scheduler is configured as a pre-emptive. In RTOS, task

execution takes place in different length and interrupts latency [7]. The present work integrates the benefits of PIC32 architecture and FreeRTOS capability to achieve a robust Data logger system.

DATA LOGGER REQUIREMENT

Among the requirements of the Micro Aerial Vehicle, Data Logger system is most important as this is responsible for logging the IMU, BMP, Magnetometer and GPS data and also the control parameters. The data logged will be used later for flight analysis.

Data logger are mainly used for the purpose such as

- Measurements over very short or very long periods of time.
- Automated data collection – no human intervention required.
- Collected data can be easily saved, appended, and exported.
- To find out the dynamic changes of the environmental parameters like temperature, pressure, altitude and control parameters etc.
- Real time data is acquired continuously within the time constraints from MAV.

SYSTEM ARCHITECTURE

The objective of the propose system is to collect data such as temperature, pressure, altitude from the sensors and log them in the Data Logger in real time. The onboard sensors such as IMU, Magnetometer and BMP will communicate with the PIC32 Microcontroller through (Inter Integrated Circuit)I2C. The GPS will communicate through UART. The RTOS will be deployed to PIC32 Microcontroller for execution of tasks with time constraints. The PIC32 Microcontroller will process the sensor data and then will store in the Data Logger. The logged data is also displayed in the HyperTerminal running in Host PC when in maintenance mode of operation.Logic Analyzer is used to measure the Worst Case Execution Time (WCET) of tasks in the schedule.

Hardware Used

The hardware requirements in this project are PIC32 Microcontroller, MPU, BMP, Magnetometer and GPS sensors.

MICROCONTROLLER

PIC32 Microcontroller was chosen as it includes 200 MHz, 2MB Flash, DSP enhanced core, 512KB RAM, 10/100 Fast Ethernet LAN8740 PHY daughter board, integrated debugger/programmer, CAN module, I2C module and UART modules. The Bus matrix is connected by two buses such as instruction and data buses separately. Bus matrix establishes the point to point communication between peripherals and also acts as the high speed switch. The internal architecture of PIC32 is shown in Figure1.

SENSORS

A sensor converts a physical phenomenon and converts it into an electrical signal. MEMS based sensors internally contain ADC for converting analog signals into digital signals. Here, sensors which are essential for the successful flight of MAV and for accomplishing a particular assigned mission are embedded into the system and to accomplish the task for the given embedded system. The sensors sense the required data and sends to the controller. In the proposed system,MEMS based sensors such as IMU, Pressure Sensor, Magnetometer and GPS are used.IMU consists of accelerometer and gyroscope and measures the velocity, orientation and gravitational forces. Information provided from this sensor is used for control and navigation of Micro Aerial Vehicle. The communication between controller and IMU is through I2C protocol. The Pressure Sensor consists of ADC, control unit, EEPROM and the piezo-resistive sensor. This sensor works on the principle of piezo resistive technology. The controller can only command the pressure sensor to initiate the temperature and the pressure measurement. The communication between controller and Pressure Sensor is through I2C protocol. Magnetometer measures the magnetic field in 3-axes which are perpendicular to each other resulting in three magnetic force vectors. It outputs magnetic heading information. It uses Anisotropic Magneto resistive (AMR) technology. The GPS device communicates with controller using (Universal Asynchronous Receiver and Transmitter)UART protocol.

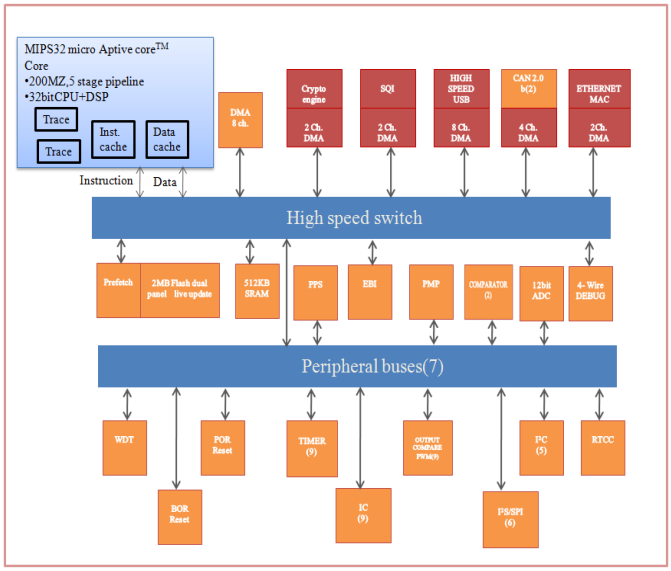

Figure 1: PIC32 internal architecture

SOFTWARE DESIGN

MPLAB X IDE is used as the Integrated Development Environment(IDE) and used in embedded applications. It is based on the Net Beans IDE. Its unified user interface can be used to integrate software and hardware development tools. The MPLAB X IDE features such as Plug-in feature extension, Powerful navigational tools,Callgraph window, Task navigator with user-defined bookmarks, Project based workspaces like simultaneous debugging sessions etc, Project Manager to create and maintain projects.

FREE RTOS

It is a free, easy to use real-time operating system which is used for scheduling. The source code is written in C. It can be ported to most of the architectures used in embedded systems. It provides services such as the task management, intertask communication synchronization, memory management, real time kernel events. Free RTOS will allow any number of tasks to run as long as hardware and memory can handle it. A task can be defined by a simple C function, taking a void*parameter and returning void. Implementation of C function will take place as tasks. Each task must return void and it mainly uses a parameter as a void pointer. Each tasks exists in states such as Running and Ready/Idle. Our application should be written as a set of independent tasks using Free RTOS and its architecture is shown in Figure 2. Task executes its each thread. The function of RTOS are such as Multiple Task creation, Memory management, Scheduling, Prioritizing the task based on the requirement etc,.

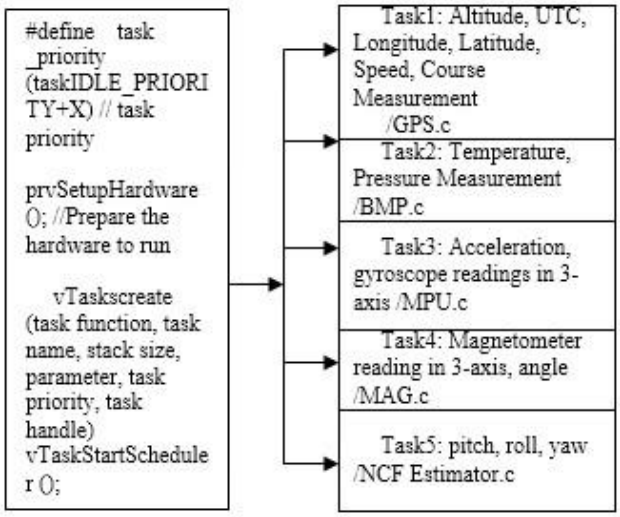

Figure 2: Free RTOS Architecture

FREE RTOS IMPLEMENTATION

Free RTOS consists of features such as portable and concise, which contains a real time kernel. It can be ported to 34 architectures.FreeRTOS can be incorporated by copying three C-source files (tasks.c, queue.c and list.c) and their associated header files into the project. These are specific to the version of the chosen microprocessor (and compiler) family. MPLAB X uses a particular folder structure and also a project navigator to keep track of all included files.

Steps to create a new FreeRTOS Project

1) Extract the latest version of the RTOS demo files from the Free RTOS website.
2) Open the RTOS Demo file in the MPLAB X IDE.
3) Add source and header files which are required to the project.
4) Change the setting (i.e., controller) in the main.c of the RTOS Demo.
5) Create a new source file.
    a) Create tasks and add it to the list of tasks that are ready to run.
    b) Inside the task create API function; pass the parameters such as task entry function, task name, priority of each task, each task size.
    c) Call the each task function in the main file.
    d) Schedule the task by using Free RTOS API function.
    e) Inside each task function include functionality of the project.
6) Build the project and debug it.
7) Display each data in Hyper Terminal using UART protocol.
8) Analyse the results using Logic Analyzer.

The applications can be split up into independent tasks and tasks share the resources of controller and switching takes place rapidly. Executions of tasks will take place in parallel which ensures the multitasking feature. Each task is created inside the main function. Inside the main function, these tasks are passed with several parameters, and they are prioritized with their own function. These tasks are in infinite loop. Free RTOS allows a number of tasks. Each task has the priority in Free RTOS. Low priority tasks are pre-empted by the high priority tasks, then low priority tasks goes into running state and then high priority task will be in the ready state, if the scheduler is configured as a pre-emptive. Several states of the tasks are defined as Running, Ready, Suspend and Blocked. In RTOS API functions are used for changing the tasks from one state to another.

METHODOLOGY

Data Logging Systems onboard MAV will record the data of sensors. In the project, FreeRTOS on PIC32 is used for development of data log system. The PIC32 Starter Kit is used for implementing the Data Logger System. The Starter kit has many peripherals support where these peripherals can be interfaced to the PIC controller with the help of GPIO pins. Individual interfacing of IMU, Magnetometer, and GPS sensors with the PIC32 is shown in Figures 3, 4 and 5 respectively. The IMU and Magnetometer sensors will be interfaced to the PIC32 Controller through I2C. The GPS sensor will be interfaced to the PIC32 Controller through UART. Initially, each sensor will be interfaced with the PIC32 Controller in order to check the sensor's working. The task for each sensor will be scheduled, in which each task after its execution will be put to sleep mode. Once individual sensors are tested, integration of all sensors with the PIC32 is done as shown Figure 6. The sensor data are fed to the PIC32 Controller. NCF [8] based Estimator is used to estimate pitch, roll and yaw. The task for reading the data from the sensors are called and once all the task finishes its read operations based on the priority assigned the tasks are repeated. The tasks are completed in the time constrained.

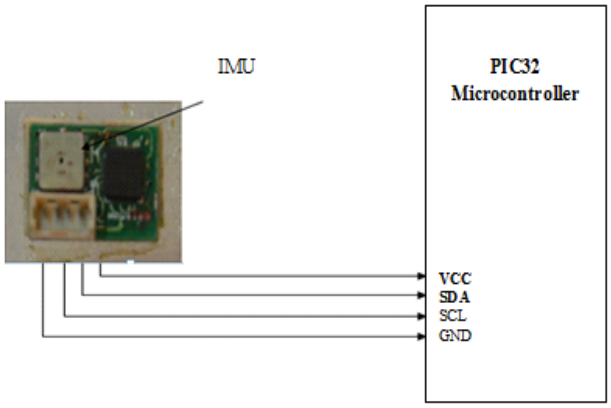

Figure 3: Interfacing of IMU with the PIC32

The task which is given the highest priority is scheduled 1st followed by the other tasks. The read time for different sensors and execution time of the NCF Estimator in PIC32 based flight controller are observed in Logic Analyzer. The sleep time for each task is also defined so that after the data is received from one sensor, the data from the other sensor is read. The tasks are called in the Main, after the initialization of I2C, UART, and the sensors. The sourcefile for the creation of tasks also consists of the priorities given for each task. A function called "vTaskStartScheduler" is called after the creation [9] of tasks so that the tasks that were created are scheduled in a continuous manner depending on the priorities assigned to each of them. MPLAB X IDE was used to code the algorithms and interact with the hardware.

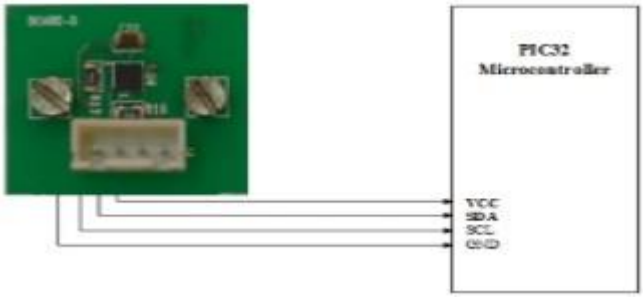

Figure 4: Interfacing of Magnetometer with the PIC32

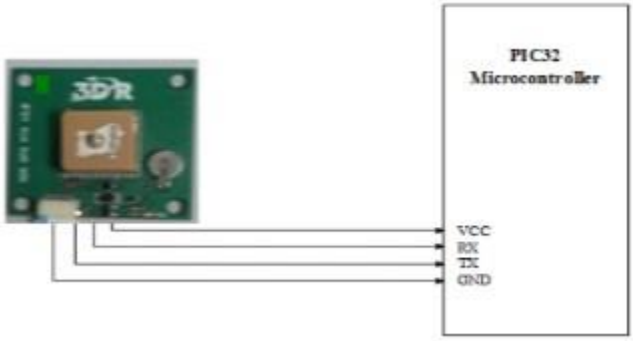

Figure 5: Interfacing of GPS with the PIC32

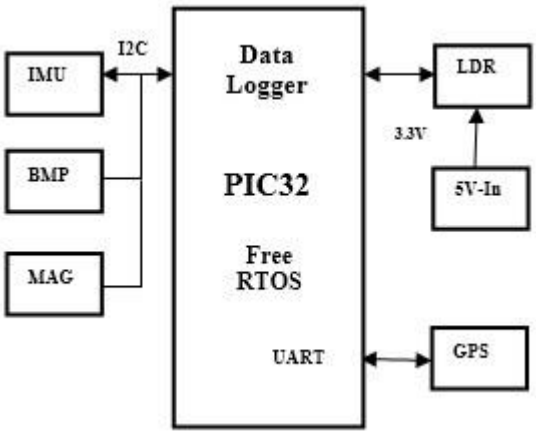

Figure 6: Data logger Block Diagram

EXPERIMENTAL RESULTS AND ANALYSIS

The experimental setup includes PIC32 Starter Kit, Inertial Measurement Unit, Magnetometer, GPS and Logic Analyzer as in Figure 7.

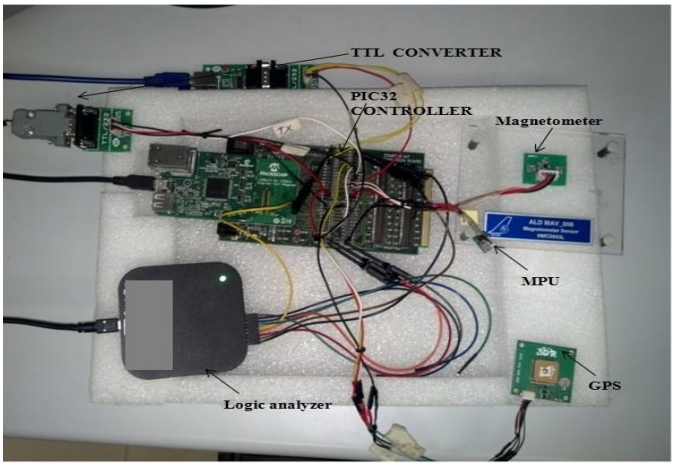

Figure7:Experimental Setup

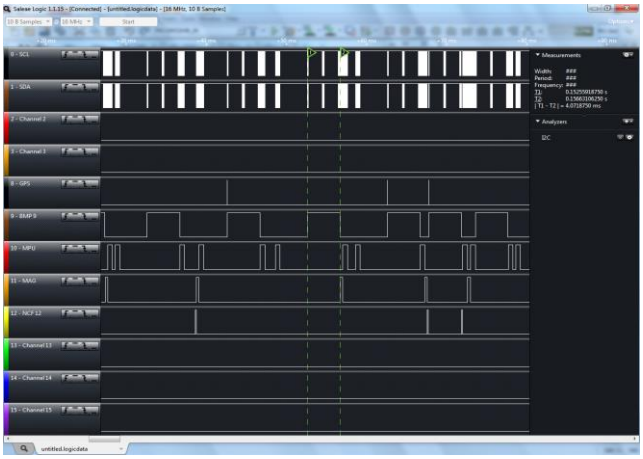

Figure 8: Read Time of different tasks seen in Logic Analyzer

All the sensors are continuously scheduled depending on the priority assigned to each task. The width of ON period for each task thus obtained represents the time taken by the task to read the data from the sensors. The off time symbolizes the sleep period of each task which means for that duration the sensor is not in operation. The ON and OFF periods of each tasks are shown in Figure 8. When all the sensors were included in a single code, the data from all the sensors could not be acquired simultaneously i.e. if we obtain the data from MPU, Magnetometer and GPS, the data from BMP could not be displayed in the Hyper Terminal thus the delay used after BMP read operation was changed to 7.95ms as the minimum time required for BMP sensor since this sensor itself would require approximately 6ms to process the raw data.

The execution time for each sensor is listed out in Table 1 and 2. The values are continuously updated as and when the data is acquired by the sensors.

Table1: Read time of Sensors

Sensor Function	Read Timings
MPU	0.49 ms
BMP	7.96 ms
MAG	0.26 ms
GPS	0.18 µs

Table 2: Execution time of NCF Estimator

Function	Execution timings
NCF ESTIMATOR	0.11ms

CONCLUSION

Interfacing of the sensors such as BMP, MPU, Magnetometer sensors and GPS to the PIC32 using RTOS is implemented. The data from the sensors interfaced are used to determine pressure, temperature, altitude, pitch, roll, yaw, orientation of the flying aircraft in real time. Tasks are created for reading the data from sensors, prioritized and scheduled continuously. The data acquired from the sensors are displayed in the HyperTerminal in Host PC.

FUTURE WORK

The complete system is to be integrated to the Micro Air Vehicle application and PIC32 used as test platform. Flash has to be interfaced and also interface with RF link and RC transmitter is yet to be done. However the challenges include high level integration of multiple sensors with flight electronics.

ACKNOWLEDGEMENT

Author would like to thank Mr. Shyam Chetty, Director NAL for continuous support and motivation. Author also would like to thank Mr. Sunil Prasad, Mr. Madhu K S,Mr. Sharath R, Ms. Shwetha and FMCD for their support in testing and integration of sensors with the controller using Free RTOS.

REFERENCES

[1] White Balki,R .,Thakur,A.,Sakhrekar,S., & Wankhede,M."Micro-Air Vehicle for Surveillance". International Conference On Advances In Engineering & Technology, 2014.

[2] Mahzan,N.,Omar,A.,Noor,S., & Rodzi,M, "Design of Data logger with multiple SD cards" IEEE Conference on Clean Energy and Technology(CEAT), 2013.

[3] Inam,R., Maki-Turja,J., Sjodin,M., Ashjaei,S., & Afshar,S, "Support for hierarchical scheduling in Free RTOS".ETFA 2011.

[4] Fairuz M.J,M.,Alabqari M.R.,A., & Shukri A,M. “Development of Interfacing system For PIC Based Data logger”.Proceedings of MUCEET2009 Malaysian Technical Universities Conference on Engineering and Technology,2009.

[5] Taylor,C., “Fusion of inertial, vision,and air pressure sensors for MAV navigation”. IEEE International Conference on Multisensor Fusin and Integration for Intelligent Systems, 2008.

[6] Mustafa,S. “A Real Time Kernel for 16nit PIC Microcontrollers”.ICACSIS., 2011.

[7] Mohideen, F, “RTOS for PIC18 microcontrollers” 5th International Conference on Industrial and information Systems, 2010.

[8] Sanketh Ailneni, Sudesh K. Kashyap, N. Shantha Kumar, "INS/GPS fusion architectures for unmanned aerial vehicles", International Journal of Intelligent Unmanned Systems,2014.

[9] Free RTOS website www.freertos.org.