ROBO-FLYER PROJECT

Unpiloted Aerial Vehicle Design.

Project Introduction & Overview

FIRST DRAFT

Hugo Vincent,
hjv15@student.canterbury.ac.nz

April 25, 2004

Contents

This document was prepared using the LaTeX 2_ε typesetting system, and was compiled with the `hyperref` package, so the PDF is hyper-linked. The TeX source file can be found in the CVS repository.

The photo on the cover is of a Multiplex Twin Star RC plane used by the Paparazzi Project as the basis of their UAVs' airframe and engine configuration.

Chapter 1

Introduction

This is a very preliminary first draft, and is being written for the sole purpose of getting some ideas on paper for discussion. It is a work in progress, so any details can (and probably will) change.

This document describes the design of an unpiloted aerial vehicle (UAV) and it's flight computer system to be developed as an open-source project. Two other open-source projects, Autopilot and Paparazzi, have similar goals – Autopilot is written for and currently runs on small helicopters, and Paparazzi on a slow, short duration electric aircraft, whereas this design is for a long endurance, medium to high altitude small airplane. Such aircraft can have a very wide flight envelope, being stable at speeds from below 20 knots (36 kmph), to above 70 knots (128 km/h). They can be launched by hand (like a paper dart or small RC model plane) and land like a glider, by skidding across a grass field (landing gear is also possible). Airplanes have a number of advantages over helicopters, including fuel efficiency, with endurances upward of 24 hours possible; they are dynamically stable, that is they can glide for long distances without control input; they are a lot quieter, and have a higher top speed. Because lift is generated by airflow over the wings, and not by continual engine input spinning a rotor, a small airplane can glide a substantial distance without power – an obvious safety benefit should engine failure occur.

The principle disadvantage of a small airplane, compared to a small helicopter is that an airplane must always be moving forward to maintain lift – a helicopter on the other hand can stay airborne (hovering) without moving, and indeed can take off and land from a very small area of ground.

Small, high endurance UAV's find uses in many varied fields, including environmental, forestry, agricultural, and oceanic monitoring, coast or border surveillance, geophysical prospecting, atmospherical and weather research and prediction, imaging for the media and real-estate industries, applications in the military, and academic aerodynamics or control research. Figure 1.1 shows a computer generated view of the Aerosonde commercial long endurance UAV (first UAV to cross the Atlantic – on a few dollars worth of gasoline!).

Figure 1.1: The Aerosonde commercial light UAV.

The design will focus on the control system for the aircraft, but an airframe design will also be presented, either in the form of an existing RC airframe or a custom design.

1.1 Design Focus & Key Criteria

The perfect design of a robotic aircraft control system would need to meet the following criteria:

Weight – Each subsystem of the aircraft must be made as light as possible. For each increase in weight of the airframe, engine or control system, the fuel capacity must be decreased or the wing lift increased.

Power consumption – The engine must consume as little fuel as possible, and the electronics must consume as little electrical energy as possible.

Reliability – For safety reasons, and to minimize the chance of a user destroying their aircraft, the plane and its subsystems must be made as reliable as possible.

Cost – To gain a critical mass of developers, the project must be made as accessible as possible; specifically it must be available to anyone, at the lowest possible cost.

Adaptability – As much of the design and software must be made as adaptable as possible to assure the project can be applied to as many different problems as possible.

Openness – To meet the above requirements, that is to decrease cost, and to increase reliability and adaptability, the aircraft will be developed under an open development model.

These criterion will therefore govern the design of the aircraft.

1.1.1 Aims for Aircraft Specifications

Parameter	Possible Values
Wingspan	∼2m
Chord	∼35cm (Aspect Ratio <5)
Engine	<2.5 BHP, ∼20cc
Dry Weight	<4kg
MTOW	5kg - 10kg
Cruise Speed	70km/h - 130km/h
Takeoff Speed	40km/h - 70km/h
Endurance	<6 hours, pref. 12 - 24hrs

Parameter	Possible Values
Range	<800km
Payload	500g - 1kg
Fuel Capacity	3kg - 5kg
Fuel Burn Rate	200g/hr - 500g/hr
Electrical Power	10W (sys.) + <20W (payload)
Navigation	GPS/inertial/baro/compass
Waypoints	Limited by memory capacity

Table 1.1: Aims/expectations for aircraft specifications and capabilities.

For lack of anywhere better to put them, expected aircraft specifications are shown in Table 1.1.

1.2 Selected Background

(To be written!) The intention of this section is to provide a simple outline of some of the relevant background, and provide references/links to more detailed information.

1.2.1 Flight: Aerodynamics, Bernoulli's Principle

- Free body force diagram (lift, gravity, thrust, drag), for static and simple dynamic flight.
- Bernoulli/Venturi Principles.
- Fluids, laminar/turbulent flow, etc.
- Airfoils, including some plots of lift vs. AOA, airspeed, chamber, thickness; importance of Reynolds Number.
- Computational Fluid Dynamics, including free (GPL) aerodynamic analysis tools such as XFOIL.
- Types of drag, relationships with airspeed and AOA; minimizing drag, optimizing lift and thrust.
- Stability
- <Some terminology will have to go somewhere around here.>

1.2.2 Inertial Sensors, Measurements and Navigation

- Accelerometers (MEMS, older non-solid state methods?)
- Gyroscopes (MEMS, piezo, laser ring, rotating flywheel, other?)
- Error and noise models of piezo gyros and MEMS accelerometers.
- Other (non-inertial) means of detecting position/attitude/velocity etc.: pitot tubes, pressure altimeter, compass and magnetometer, gravity reference, horizon detection.
- Terrain Navigation, VFR, vs. Instruments,
- Route planning, available information (terrain maps, weather forecast...)

1.2.3 The Global Positioning System

- Theory of operation, etc.
- Receivers
- Error and noise models.

1.2.4 State Estimation: The Kalman Filter

Inertial measurements have to be integrated over time to estimate position – but because of this integration, small errors in measurement are quickly amplified. On the other hand, the inertial measurements are very accurate over shorter time frames, and are capable of high update rates. By comparison, measurements from a GPS receiver, can have relatively large short term fluctuations or noise, but are very accurate over time. GPS receivers are typically capable of only a relatively slow update rate, typically 1Hz. Thus, GPS and inertial measurements need to be combined in such a way that they compliment each other.

The technique almost exclusively used for combining inertial and GPS position measurements is known as the *Kalman Filter*. The following introduction to the Kalman filter is from [2]:

> Theoretically, the Kalman Filter is an estimator for what is called the linear-quadratic problem, which is the problem of estimating the instantaneous state of a linear dynamical system perturbed by white noise – by using measurements linearly related to the state but corrupted by additive white noise. The resulting estimator is statistically optimal with respect to any quadratic function of estimation error.
>
> Practically... it has enabled humankind to do many things that could not be done without it, and it has become as indispensable as silicon in the makeup of many electronic systems.
>
> To control a dynamical system, you must first know what it is doing. For these applications, it is not always possible or desirable to measure every variable that you want to control, and the Kalman filter provides a means of inferring the missing information form the indirect (and noisy) measurements. The Kalman filter is also used for predicting the likely future courses of dynamical systems that people are not likely to control, such as the flow of rivers during a flood, the trajectories of celestial bodies, or the prices of traded commodities.

More mathematically, the Kalman Filter is ...

1.2.5 Control Theory: Feedback & PID Control

- Some maths will have to go somewhere
- Feedback, Feedforward.
- PID
- Dealing with non-linearities, noise, offsets, etc.

Chapter 2

Airframe & Engine

2.1 Airframe: Fuselage, Wings & Tail

The airframe will initially take the form of a conventional 3-axis airplane, built as a medium-sized, radio-controlled, model airplane. The exact type of airplane to be used has not been decided, but it is estimated that it would require a payload capacity of 1kg or less for the electronics and cameras, in addition to the weight of the engine, servo's, fuel (existing + extended capacity), batteries etc. The sensors and electronics would weigh less than 300g, so the airframe would have plenty of capacity for small sensor or imaging payloads, weighing on the order of 200g to 700g. Eventually a custom airframe will be required, as the requirements of a UAV are somewhat different to the requirements and wishes for a hobby model aircraft. Model aircraft are not normally designed for endurance, or operation at altitude, focusing instead on a realistic looking miniature airplane appearance (of course there are *exceptions*, as always). The design of a highly efficient, stable, low drag, airframe is one of the eventual key goals of this project, and the aircraft performance aims will not be met without it.

It is expected that the wingspan will be about 2m, and a takeoff weight (including fuel) of about 5kg to 10kg. Most estimations point to a dry weight of around 3.5kg to 4kg. For legal and regulatory reasons, it is desirable to keep the total weight under 15kg for operation in New Zealand as a RC model (in some other countries the limit is 5kg, but to get it under 5kg might involve either nasty compromises or greatly reduce fuel capacity).

Pusher propellors (rear facing) are apparently more efficient (by reducing drag at the front of the plane) so this configuration would be preferred. A pusher prop matches an inverted-V tail configuration very well, and many existing small UAVs take this approach (including the Aerosonde shown previously in Figure 1.1, and the INEEL research UAV shown in Figure 2.1). The INEEL research UAV also uses a blended wing-body design, which can reduce drag and increase efficiency, as well as allow more room for electronics or fuel. A blended wing-body design would not increase fabrication difficulty or cost much, if the correct construction technique was used (see section 2.1.1).

Figure 2.2 shows the STAR (Society for Technical Aero-model Research) TAM26 – Trans-Atlantic Model number 26. TAM26 is the smallest aircraft to have crossed the Atlantic – a feat which took about 2 days. The wingspan is about 1.9m, and the complete plane weighs only 4.99kg including about 2.3kg of fuel. The engine (based on a commercial RC model engine) and the fuel system were specially modified for high fuel efficiency.

Some preliminary concept design work has been completed in SolidWorks, an easy to use 3D mechanical CAD software package that includes plugins for aerodynamic analysis and structural stress analysis. The exterior surfaces of the wing, fuselage, and tail have been drawn, based on some rough calculations, and will be honed over time. A 3D rendering and 3-Views [will be] shown in Figure X, [and more details will be added as work progresses].

2.1.1 Construction Techniques

The wings will be constructed from fiberglassed expanded polystyrene foam, an increasingly common RC model technique. The wings will be joined and strengthened with a fiberglass or carbon fiber tube through the fuselage, and the fuselage will blend with the wings. The fuselage will also be constructed from foam and fiberglass, but at this stage, the details are unclear (the lost-mould process is the most probable). Much more design work needs to be completed before we get to this stage.

The main advantages of fiberglass-foam construction, are ease and precision of construction (especially for wings), and excellent durability, but it can result in a fractionally worse weight to strength ratio than traditional

Figure 2.1: A research UAV used by the INEEL Autonomous Flight Group, sponsored by DARPA.

balsa-wood RC model construction.

2.2 Engine, Propellor, Fuel, & Generator

The engine would be selected according to fuel efficiency, weight, noise/vibration, and cost. So called "glow engines" are common in the model aircraft world; they typically run on a mix of methanol, oil and nitromethane (some can run on a diesel-based mixture). The engines are noisy (running at up to 15,000rpm), unreliable, and fuel inefficient, but they are light weight and low cost. For a long endurance flight, these engines are obviously out of the question due to the poor fuel efficiency. Two-stroke gas engines are also popular – they run on gasoline (cheap), they are reliable, relatively inexpensive, and relatively light, but are quite noisy and not particularly fuel efficient. The best engine for fuel economy would be a 4-stroke gasoline engine – unfortunately, most such engines designed for model aircraft are much heavier than similarly powered 2-strokes, or glow engines. Probably the best compromise would be a 4-stroke glow engine, such as the O.S. Engines FS-70S*II*, modified for electronic ignition instead of glow (this is the approach taken by the STAR group for their TAM transatlantic flights, Figure 2.2). Other options include 2- and 4-stroke diesel engines, and modified consumer engines (e.g. weedwacker or chain-saw engine).

An interesting new development in the RC engine world is electronic fuel injection (EFI) – at least two manufacturers sell engines utilizing an EFI system instead of a carburetor. The OS Engines 1.40RX-FI (shown in Figure 2.3) is one such engine, delivering 3.5 HP from a 2-stroke nitro glow cycle, and weighing 810g including the engine management computer and muffler. The flight computer is separate, and could possibly be eliminated or modified, so that the engine could be run from a MCU on the flight control board. Another consideration is the engines ability to operate at altitude, and it's performance and efficiency at higher altitudes (hopefully the airplane will be useable above 10,000 feet). Two model aircraft engines, a 4-stroke nitro glow, and a two stoke gasoline engine are shown in Figure 2.3.

The propellor is extremely important in obtaining good thrust, while minimizing drag. On "real" airplanes, a

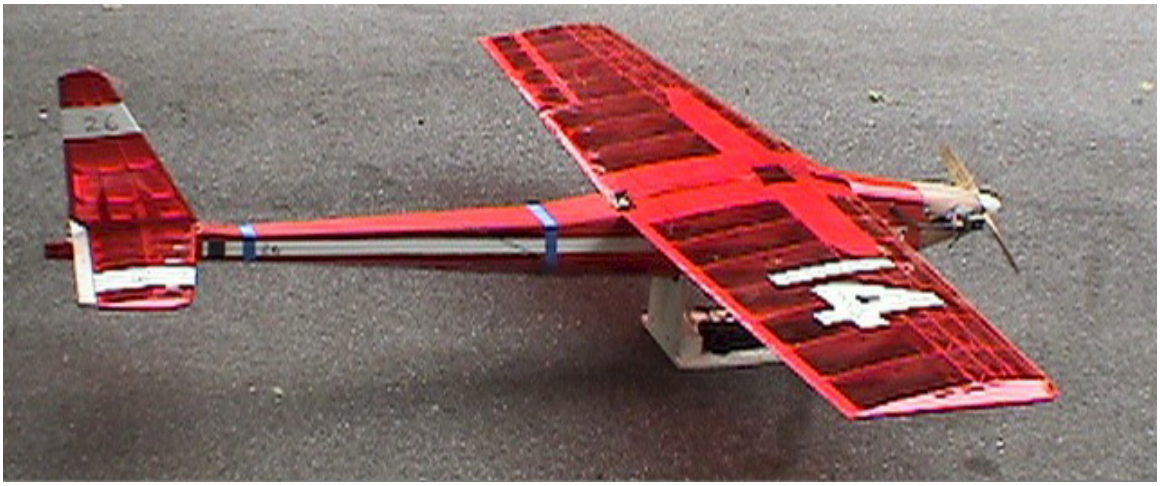

Figure 2.2: The STAR group's TAM (Trans-Atlantic Model) 26 UAV.

Figure 2.3: *Left:* Enya 53.4C 4-stroke 1.0 BHP glow engine. *Center:* Zenoah GT26 Petrol 2-stroke 3.5 BHP engine.*Right:* OS Engines 1.40RX-FI electronic fuel injected glow engine.

so called 'constant-speed' propellor is normally used, that is a propellor that can adjust its pitch (while in flight) to change thrust, while staying at at the same rotational speed. Such propellors allow for better efficiency, but unfortunately, are too complex and heavy for use on a small UAV. As a result, a 'constant pitch' propellor must be used, that is one that has a fixed pitch, so thrust must be changed by changing the rotational speed. A result of this is that for top airspeed, the engine must be running at full throttle, where the fuel consumption is highest, even though all the torque available may not be utilized. The best solution is to use a propellor with a high pitch angle, to get more thrust at lower rotational speed, by generating more torque; the disadvantages of this is that the minimum thrust, and thus minimum straight-and-level speed is higher, the idle speed must be higher to generate the torque, and the acceleration and climb performance is worse. Comparison with other planes of this size and type (such as the Aerosonde and the TAM), indicates that the optimal propellor will be about 14-12[1].

A small generator (or alternator) will be run from the engine to power the electronics and computers, decreasing dependence on batteries – fuel has a much better energy to weight ratio than batteries of any type. Small generators designed for RC planes are available, that also double as a starter motor. About 30W would be an appropriate generation capacity – such a load would have little impact on the engine (equating to about 0.07 HP, assuming 50 % conversion efficiency). Any excess electrical energy could be dumped into electric heaters to prevent condensation or icing in the electronics, camera optics, and carburetor (at 10,000 feet, the "standard"

[1]For model airplane propellors, the first number is diameter of the prop in inches, and the second number is the distance (in inches) that the prop will travel in 1 revolution in a perfect fluid. Studies indicate the best props can achieve about 80-85% efficiency in air.

atmospheric temperature is -6^{o}C, and at 20,000 feet it is -28^{o}C).

Another possibility is for an electric propulsion system – an electric motor driving the prop (perhaps through a gearbox), and a large battery pack. Such a system would be very nice, in that it would be quiet, clean, vibration-free and reliable (and electrical engineers can generally work with them better than we can with gas engines); but with todays battery technologies the aircraft would have shorter maximum duration (1.5 hours at the *very* most).

An small foam RC airframe with an electric motor for propulsion has been selected for developmental use, as it is cheaper and easier to work with, and also substantially more resistant to "accidental landings". It weighs about 1.4kg including batteries, and allows for a payload of about 200g.

Chapter 3

Flight Electronics

For best overall performance, power efficiency, size and weight, a custom circuit board (PCB) would be needed. The rest of this section discusses the major parts of the custom design. Figure 3.1 shows an overview of the hardware platform.

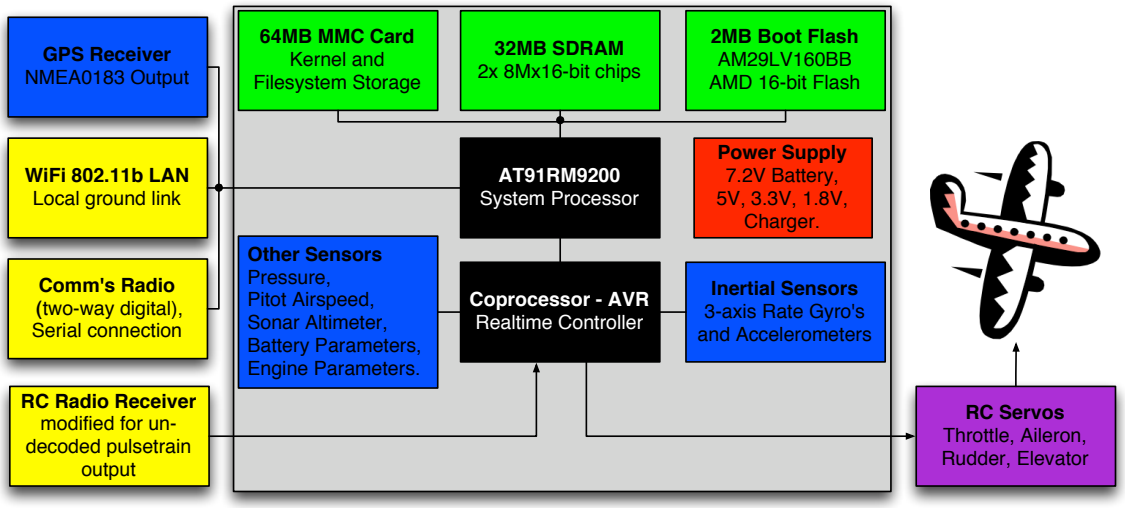

Figure 3.1: Simplified Block Diagram.

In the figure, the grey box represents the new circuit board to be designed, containing all the electronics (except for the communications modules and the GPS receiver) and most of the sensors needed by the system.

Another option is to use a pre-made board – one possibility is the open-source royalty free Linux-On-Module-ARM-9, which (in conjunction with a relatively simple, custom daughter-board) would contain all the major parts of the hardware part of this project. A photo of the LOM-ARM-9 is shown in Figure A.3 in the Appendix. The possibility of using an existing system board will not be discussed further in this document – the rest of Chapter 3 focuses on the requirements and initial design decisions of a custom-hardware approach[1].

3.1 System Processor

The main flight processor has been chosen to be the Atmel AT91RM9200 [1], which features a 200MHz ARM920T (16kB I-cache, 16kB D-cache) core, and a large selection of useful peripherals. The processing throughput is

[1] It is expected that the first prototype(s) will be made using the LOM-ARM-9, with a custom daughter board containing the AVR, all the sensors, the power supplies, interfacing circuits, and any miscellany. With careful design of the daughter board, little work would be required to move to a full-custom design.

enhanced by the large caches, which allows the memory bus to be used for DMA memory transfers while the CPU is working from cache – Atmel specifies that 200 Dhrystone MIPS can be expected with a core frequency of 180MHz. It is available in small quantities though the normal channels at a relatively low cost, and comes in a 28x28mm PQFP208 package (i.e. hand solderable; not BGA).

According to the manufacturers datasheet, the CPU core in addition to a variety of peripherals will use about 30mA at 1.8V and about 25mA at 3.3V (Total: $\leq$150mW). This is considerable less than, for example, the Intel XScale PXA255 at full bore ($\approx$1W).

The processor features 16kB of on-chip full speed SRAM, an on-chip boot ROM, EBI, MMC card controller, USB host and device ports with on-chip transceivers, an ethernet MAC, I^2C, SPI, four USARTs, 3 SSC (e.g. for an I^2S audio codec), 3 general purpose timers, a system timer unit and realtime clock, 8 interrupts, up to 122 GPIO's, JTAG ICE, and a debug UART. The block diagram of the processor (from in the datasheet) is shown in Figure A.5, in the Appendix.

3.2 Memory

The main memory for the system would be 16 or 32 MB of SDRAM, soldered onto the system board. This would be used for both data and executables. For persistent storage, the system would include a Multimedia Card (MMC) socket.

For booting, a 2MB or 4MB 16-bit wide parallel flash device would be included (probably an AMD or Intel part), and would contain a power-on self-test (POST) routine and a Linux bootloader (U-Boot). The actual kernel could be stored on the MMC card, along side the filesystems.

The main software (Linux kernel) would be loaded off an MMC card (by U-Boot) into the SDRAM for execution. The linux root filesystem would reside on the same card – thus simplifying development, as to install an entire new system (except the bootloader), you could just swap the card. The MMC card would also have lots of room to store a very detailed flight log, maps, photos, and so on.

Another option would be for the processor to boot off of a serial flash memory, such as the Atmel AT45DB041B 512kB DataFlash, which is available though normal distributers in small quantities, and in a hand-solderable SO8 package. A special feature of the AT91RM9200 processor is the on chip bootloader ROM, that can load a software image into on-chip SRAM via the SPI or I^2C ports. The bootloader ROM also supports downloading an image to SRAM from a PC attached to the serial or USB device ports, and Atmel supply PC software to use it.

3.3 On-board peripherals and communications.

The processor has 4 UARTS that would be used as follows: one to communicate with the realtime coprocessor (the AVR), one to receive the NMEA sentences from a GPS module, one "spare" to talk to a payload module, and one for a radio modem (or some other means of talking to the ground base station). The processor has a separate dedicated debug port that can be used to print debug or error messages and upload new software.

Depending on how much spare CPU time will be available while the control, navigation, and communication processes are running, an a software modem may be added. It would use an I^2S audio codec, and a voice grade analog radio unit. Suitable codec's might be the Texas Instruments PCM3501 or PCM3002, both of which come in a 24-pin SSOP. On the ground station, the software modem could run through a PC sound card like what has been done here: http://www.raag.org/sv2agw/agwsc.htm.

The USB device port would be wired up, but probably not used for anything other than the processor's bootloader software download capability. The USB host port would be connected to either a USB WiFi 802.11b module for ground comm's, or a USB web-cam for image capture. Using USB rather than CompactFlash or PCMCIA requires much less on board logic (its all on-chip) and will make the PCB much simpler and cheaper. The Linux operating systems has many existing drivers for USB peripherals and the USB host, so the software effort shouldn't be too difficult.

The ethernet MAC would probably not be wired up to a PHY, transformer and RJ45 socket, due to the high power consumption, and relatively large amount of board space that they would require. Obviously the ethernet would only be of use on the ground (pre- or post-flight), so there is no good reason to include it.

3.4 GPS Module and Sensors

Any GPS module that can output position and velocity measurements via a serial port, preferable in standard NMEA sentences would suffice. Suitable modules include the Rockwell/NavMan Jupiter TU30-series – these can

sometimes be found on the surplus market at attractive prices. A photo of the Navman Jupiter Pico is shown in Figure A.2 in the Appendix.

A digital temperature sensor would be included on the main board to compensate flight sensor measurements and power management functions to the current ambient temperature. This is necessary because most of the flight sensors used have considerable drift over the temperature range expected in a medium-altitude flight. Also, engine performance and battery characteristics may be affected by temperature, and therefore need to be managed accordingly.

All the flight sensors: Gyro's, Accelerometers, pressure sensors (for altitude and airspeed); and engine management sensors (temp, tacho, etc.) would be attached via the realtime coprocessor.

3.5 Power supplies and Reset

The system will have relatively low current draw on the 1.8V rail, but more considerable current draw on the 3.3V rail (SDRAM, GPS module). The 5V rail is used to power the realtime controller, sensors, analog components, and USB device (such as a WiFi card) – it therefore needs a reasonable current capability, several hundred milliamps.

The main system power will probably be a 6 or 7.2 volt NiMH or LiIon battery pack with about 2000mAh of capacity. The 5V rail can be derived by an LDO linear regulator directly from the battery, the 3.3V rail would be supplied by a switching step-down converter, such as those available from Maxim, National Semi, or Linear Tech. The 1.8V rail would be derived from the 3.3V rail by an LDO regulator. Another switching regulator will probably be used to provide a regulated power supply for the servos. Alternatively a separate battery might be used.

Circuitry to regulate and clean the output of the generator will also be on board, including the ability to charge the main batteries.

The ADC in the AVR will measure battery voltage and servo current, to give an estimate of battery life with the current power draw, and also judge whether the engine is throttled high enough for the generator to work properly.

To reset the various components on the board at power on, or during a power supply brownout, an integrated reset controller may be used, such as those from Maxim or Microchip. It can be a difficult to obtain such devices in small quantities, so other alternatives may have to be considered. An interesting possibility is to use the AVR to control the reset of the main ARM processor.

3.6 Realtime Coprocessor

A separate realtime controller is useful to offload all the interrupt-heavy timing and measuring functions from the main CPU (a large operating system such as linux incurs a substantial overhead when servicing interrupts). It is also much simpler than the main processor, and therefore somewhat less likely to crash, thus improving the overall reliability of the system.

The realtime controller design would be similar to that of the existing Autopilot Rev 2.4 AHRS board, and would run some of the same software. In my opinion, it would be better to have the realtime controller act as a sort of intelligent data gathering agent that can provide the system with sensor data, and alert it to critical problems (e.g. engine overheating) via an interrupt, but leave the actual processing (Kalman and PID filters) to the main processor.

3.6.1 Microcontroller

The Atmel AVR microcontroller has been selected, because of it's speed, flexibility, ease of use, and software base (especially GCC). It is an 8-bit RISC CPU, clocked at 16MHz, with a throughput approaching one operation per cycle (16 MIPS). All of the AVR's have on chip flash and SRAM for program and data storage respectively. Most of the ATmega series feature 10-bit ADC's, with 8 or more inputs, a variety of serial ports, timers, and other peripherals.

The ATmega16 in particular was chosen – it includes 16kB of flash, 1kB of SRAM, and 0.5kB of E^2PROM memory, serial interfaces, PWM outputs, a RTC, and an 8-channel ADC (2 of the channels have differential inputs and programmable gain amplifiers). It comes in a 40-pin PDIP or a 44-pin TQFP package, and draws a current of about 20mA at 5V when running at 16MHz. In addition, the ATmega16 has a JTAP TAP (test access port), allowing for in-system programming, debugging, and testing.

3.6.2 Inputs; Sensors

To fly an aircraft autonomously, you need to know the attitude (angle with respect to a reference frame), position, velocity and acceleration. To measure attitude directly is not possible without an external reference (such as the horizon, the earths magnetic or gravitational field, etc.) so to avoid dependencies (e.g. on the weather, for a horizon detector) changes in rotation are measured and integrated. The rate of rotation is measured in each of the axis's with a solid-state gyroscope, such as those manufactured by muRata, Tokin, or Analog Devices. The acceleration is measured in 3 axis's with micro-machined accelerometers (ADXL202 etc.), and used to compensate the other measurements if they are made in an accelerating frame, and to give a attitude reference (measuring the earths gravity) when the aircraft is not accelerating (accelerating also includes turning, not just speeding up or slowing down). Because a airplane must always be moving forward to fly, the GPS module can be used to provide compass functionality (by computing the difference between successive position measurements) – a magnetic compass is therefore not needed. The position and velocity is supplied by the GPS module.

A barometric (pressure) altimeter and airspeed indicator would also be included because they provide useful and necessary information to the flight controller. An airspeed indication is necessary to fly in air (that may already be moving, due to wind), because lift is generated by air velocity over the wing airfoil – for example if you are flying with a 15 knot tailwind, and the GPS indicates a velocity of 25 knots, the speed of air over the wing is only 10 knots. Pressure altimeters are useful to either measure altitude, or to measure air pressure by comparing the pressure with the GPS indicated altitude. An external temperature sensor would also be useful to provide a more detailed picture of the outside air.

Several sensors will be needed for engine management: tachometer, CHT (cylinder head temp.), EGT (exhaust gas temp.), fuel level. I am considering the possibility of another small MCU such as the ATmega8 to be used solely for engine management.

If automatic landing and takeoff is desired, a ground referenced altimeter is needed, typically a sonar range-finder facing downwards. Such altimeters can only operate below about 20 to 30 feet above the ground, but for a slow moving airplane, that is probably sufficient for a controlled landing flair.

I am always open to other flight sensor possibilities, including infrared thermopiles, optical horizon detectors, etc. Other amateur UAV groups (e.g. the Paparazzi Project) have reported good success in attitude determination by the use of such sensors.

3.6.3 Outputs; Servo controller

The airplane's control surfaces and engine throttle are actuated by RC modeler's servos. Such servo's pack a motor, gearbox, feedback, and an electronic controller into a very small, robust, reliable, low cost package. They are available in a number or types specifically designed for small model aircraft.

Servo's accept a control input in the form of a pulse train, where the pulse width is proportional to the desired position of the output shaft. The standard is a pulse of width between 1ms and 2ms, repeated ever 20ms. This translates to a output position between 0^o and 180^o. The electronics integrated into the servo moves the output shaft and maintains its position by a closed feedback loop, but only when the pulse train is continuous – if you stop outputting pulse data, the servo will drift.

An airplane requires at least 4 servos for flight: elevators, ailerons, rudder, and throttle; on certain airframe designs, more might be needed for flaps, spoilers, variable-pitch propellors and so on. One or more spare servo controller channels are needed to support payloads such as a pan/tilt/zoom camera. Further, many airframes would use multiple servos to accomplish one function, for example have one servo in each wing to operate the ailerons. Eight is a good minimum number of servos to support. The AVR's timer/counters will be used to generate the output pulses. Software and hardware designs exist to do this in the Autopilot package.

A manual over-ride capability is required for safety and development. To this end, a PPM/PCM RC decoder will be included in the AVR software, directly coupled to its servo outputs, and switched between manual and automatic control by one of the transmitted servo channels. Another possibility is to use a small GAL or PAL, like the Lattice ispGAL22LV10 – advantages would be an even more robust fail-safe, but it comes at the expense of another IC, and its associated board space.

3.7 Communications

It is expected that the plane will include 2 or more communications channels back to the ground station; several possibilities have been identified. The first is the 802.11b WiFi Wireless Ethernet link – it has high bandwidth, but short range: good for imagery (optional). The second is a VHF or UHF modem (with a speed of at least ~50kbit half duplex): good for control and telemetry over a much longer range. One possible radio modem, the

XStream 900 is shown in Figure A.4 in the Appendix. Other options may include a satellite radio, a dedicated video/TV transmitter, or a cellular phone modem.

A RC model radio receiver is also required for safety and development – some such receivers are very small and consume minimal power.

Chapter 4

Software

Figure 4.1 shows how the various processors are run, as to which operating systems or virtual machines are used.

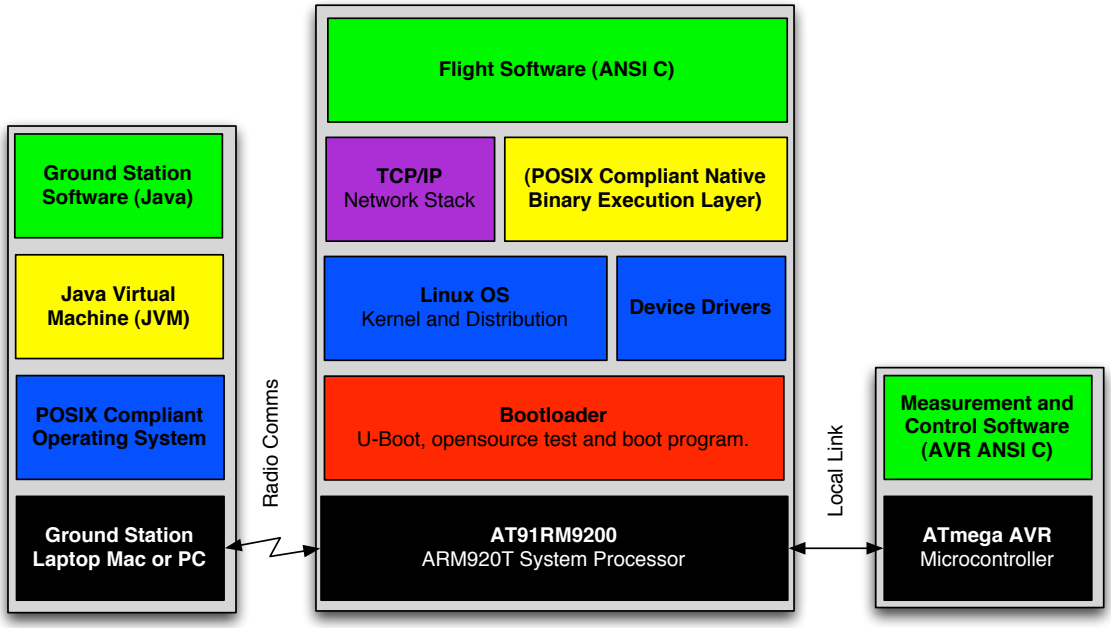

Figure 4.1: Software overview.

The main processor would run a Linux kernel and distribution with a set of normal linux tools, such as those provided by BusyBox. Linux provides many useful advantages for a system like this UAV, including a robust network stack, USB stack, filesystem support, protected memory and multitasking/multithreading, and a very convenient programming environment.

An open-source bootloader, U-Boot, will be used – it includes POST (power on self test) routines, flash updating routines, MMC and filesystem support (allowing the kernel to reside on the MMC card) and, of course, a Linux boot function.

The main processor will run several processes to allow the plane to fly. These would include the following. An AHRS (attitude and heading reference system) process to regularly receive and combine the gyro and accelerometer data via a Kalman filter to give a state estimate. The GPS data position and velocity estimates would also be integrated into the state estimate at this stage. Such an approach would be called a loosely-coupled inertial measurement system. Most of the software for this part of the project already exists, as part of the Autopilot project, or elsewhere.

The flight controller and navigation process would compare the measured attitude, position, velocity and heading with the desired values and change the control parameters to actuate a change. This control loop will probably take the form of a modified closed-loop PID (proportional, integral, derivative) controller, with the addition of feedforward state prediction.

The flight planner process would accept input from the user or a prerecorded flight plan and generate a series of waypoints in-between the two end points. It would implement a variety of common aviation rules and take into account factors such as the wind, and possibly prerecorded or live weather forecasts (to fly around storms - remember this is designed as a long endurance UAV), or terrain information (so as to fly around rather than into hills).

Small engines can achieve substantially better performance and fuel economy though tight monitoring and control. Indeed many commercial UAVs have specially built computerized engines with tightly integrated electronic fuel injection and electronic ignition. Maybe that is something to aim for.

Recording of flight and atmospheric parameters throughout the mission would be necessary for tuning the aircraft control algorithms, and for keeping a log of where the craft went and what it was going next at any stage of the mission (like a "black box" on commercial passenger aircraft).

The payload, such as an imaging or sensor package may require control, communications or recording facilities, which would primarily be provided by the main processor.

Finally, a method of accepting control input from the ground, and transmitting progress and status reports is of obvious importance - it would need to operate over any or all of the available communications channels, include rigorous error checking and correction, and allow for unexpected circumstances.

It is expected that the software package will leverage a large amount of the existing Autopilot code base, as well as the other open-source UAV projects, but obviously a large amount of new code will be necessary.

4.1 AVR Software

The AVR microcontroller would run control and measurement routines, structured as a sort of "Virtual Machine", with simple, atomic instructions from the system processor being used to control it. The virtual machine architecture was inspired by the approach taken by the BitScope project (see the article in Circuit Cellar Magazine Issue 97 for more details). The idea behind this is that the flight software on the system processor can command the AVR to do relatively arbitrary functions without writing new AVR firmware. Many of the commands would be higher-level functional commands, such as "set servo 2 position", "read ADC channel 1", etc., but there would also be low level operations for reading and writing to peripheral registers and accessing the hardware. (It would most probably be desirable to protect registers that control vital functions such as servo control or the ADC.)

A virtual machine approach would allow for different applications of the board, by simplifying customization to suit the special features of the aircraft (such as still camera's or landing gear) without having to rework the AVR low-level software.

4.2 Ground Station Software

The ground station software would run on a normal laptop, running a Unix-like operating system, and a Java Virtual Machine. Communications with the aircraft would be made through a radio modem connected to a serial port, and through TCP/IP over the WiFi link.

It is expected that the serial communications channel would be managed by a daemon process that would abstract the low level interface, and provide a higher level link to the Java flight software, perhaps through sockets or TCP/IP.

Appendix A

Miscellaneous Material

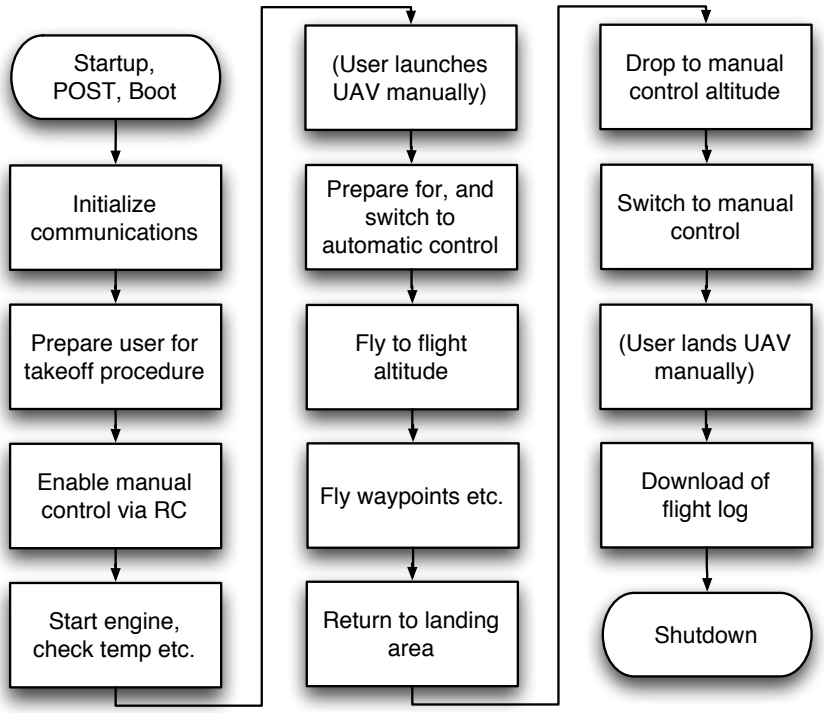

Figure A.1: Flow chart showing the stages of an automated flight.

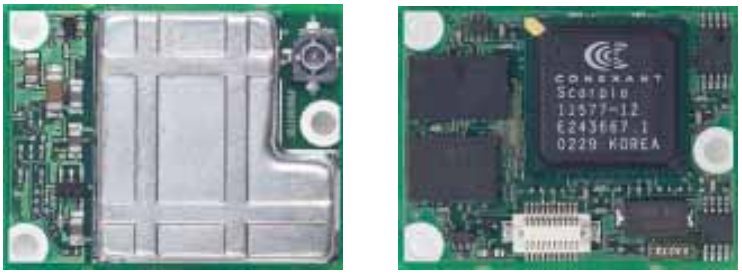

Figure A.2: Photos of the NAVMAN Jupitor Pico GPS Module. It measures ∼35mm on the longer side

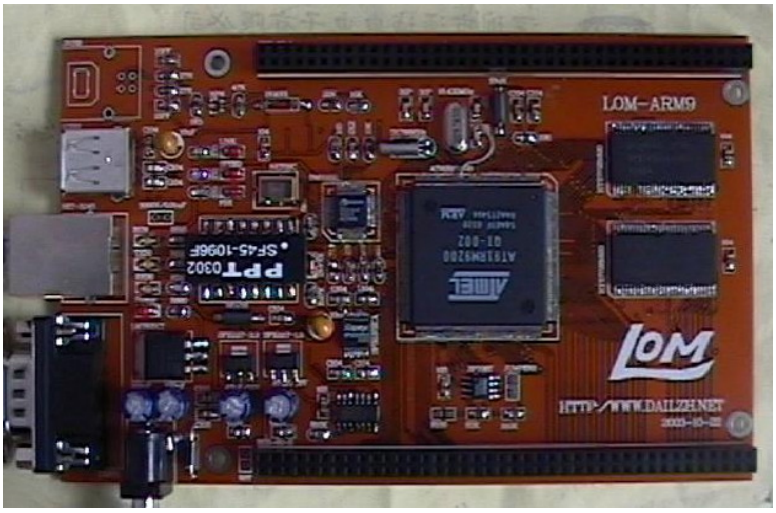

Figure A.3: Photo of the Linux-On-Module-ARM9, which is based on the AT91RM9200.

Figure A.4: Photo of the XStream 900MHz radio modem module.

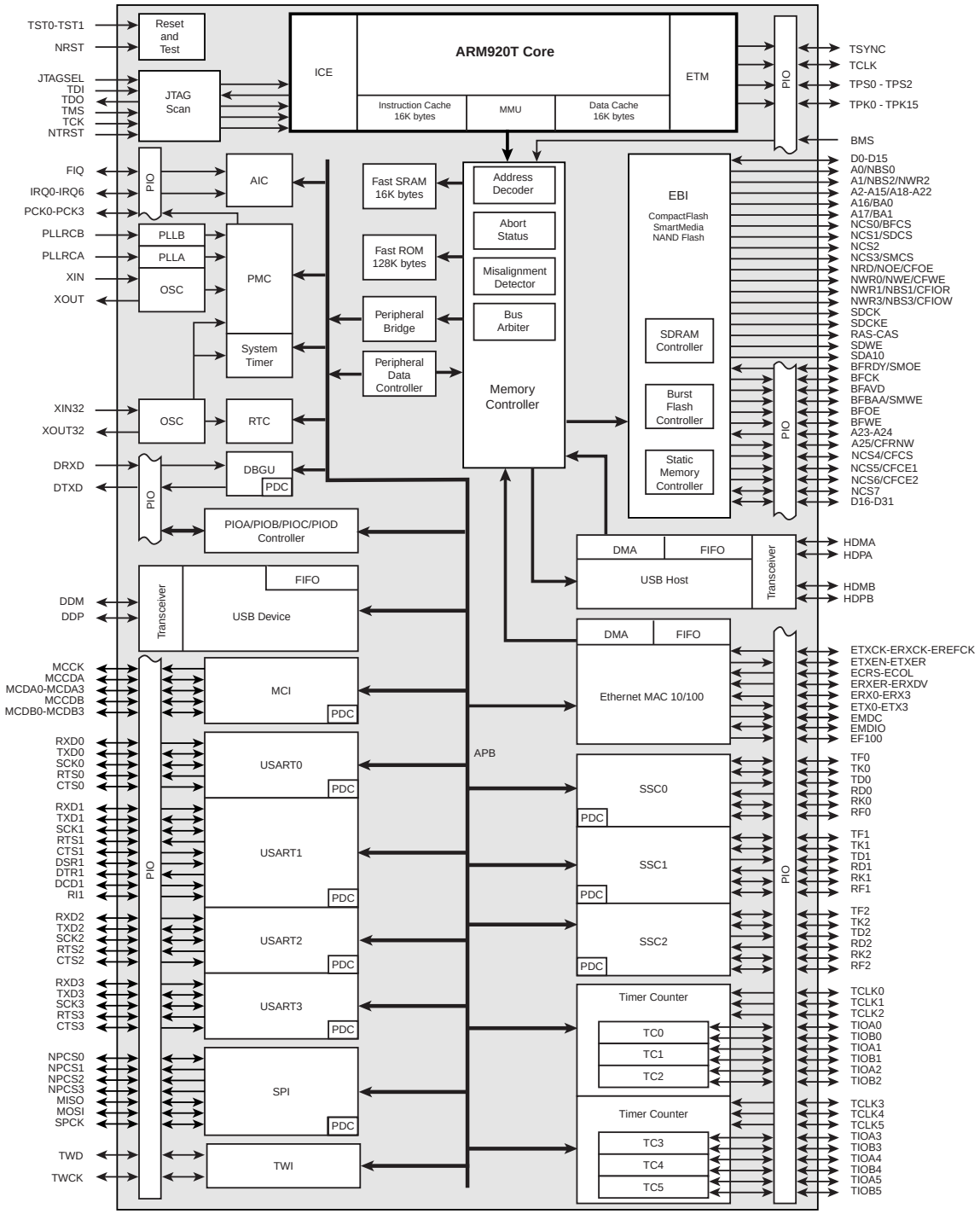

Figure A.5: Block diagram of the Atmel AT91RM9200 microcontroller [1].

Bibliography

[1] Atmel, *AT91RM9200 ARM920T Microcontroller Datasheet*, USA, 2003.

[2] Grewal, Mohinder S. & Andrews, Angus P., *Kalman Filtering: Theory and Practice Using MATLAB*, 2nd Ed, John Wiley and Sons, 2001.

[3] *(Remaining references to be resolved later.)*