TU Delft	**AE4265** **Group Assignment Description**

Course:	AE4265 Space Embedded Systems
Course Responsible:	Dr. Alessandra Menicucci
Project title:	Design of a fault-tolerant on-board computer for an interplanetary small satellite mission
Keywords	Embedded systems, Space Radiation, fault tolerant, Cubesat
Summary	In this assignment the on-board computer

Introduction

The rapid performance advance in modern electronic components targeted at the mobile and IoT (Internet of Things) market is pushed by pervasive presence of connected electronic devices in many areas of our lives. Thanks to the continous work of the technical and scientific community aiming at enhancing the reliability and availability of devices and systems, COTS based embedded systems are becoming intersting solutions for space applications. In particular in small satellites missions we can find a large number of examples of embedded systems use. Small satelites (e.g. Cubesats) have been so far only launched in Low Earth Orbits but the future projects aim at interplanetary missions for which the level of reliability of the current architectures needs to be enhanced.
The general trend is to have future space avionics changing from centralized intelligence to distributed autonomous functions, thanks to the availability of high capacity FPGAs and microcontrollers that offload tasks alternatively concentrated in the on-board computer. This will allow higher reuse of building blocks, especially IPs, and a lower SW footprint, but guaranteeing a similar (or better) system level availability and reliability needs an in depth study of industrialization of these new architectures, especially from SW verification and validation point of view.
Software based data acquisition/processing and simple control applications are widely used in many spacecraft subsystems.
They allow implementing software based control architectures that provide a higher flexibility and autonomous capability versus hardware implementations. For this type of applications, where limited performances are requested to the processor, general purpose microprocessors are usually considered not compatible due to high power consumption, high pin count packages, need of external memories and peripherals. Low-end microcontrollers are considered more attractive in many applications such as:

- propulsion system control

- sensor bus control
- robotics applications control
- stepper motor control (e.g. Solar Array Drive Mechanism)
- mechanism control
- power control
- particle detector/radiation environment instrumentation
- thermal control
- antenna pointing control
- AOCS/GNC (Gyro, IMU, MTM, Sun Sensor)
- Remote Terminal Unit (RTU) control
- Simple instrument control
- Wireless networking
- Telecommand Decoding

In these kind of applications the microcontroller device should have a relatively low price, a low power consumption, a limited number of pins and must integrate small amount of RAM and most of the I/O peripherals for control and data acquisition (serial I/Fs, GPIO's, PWM, ADC etc.). The requirements for memory and program length are usually minimal, with no or very simple operating system and low software complexity.

Scope of the assignment

In this group assignment a **microntroller-**based design for a safety mission-critical function in a miniaturized satellite will be developed.

Sub-projects

Four baseline sub-projects with 'closed control loop' structure can be implemented. Each group will choose one between the following:

1) Autonomous Thermal Control
2) Flexible Telecommand Decoder
3) Intelligent Sun Sensor electronics
4) Solar Panel Drive Mechanism

1: Autonomous Thermal Control

What You Will Need

- Arduino/ARM development board
- Temperature sensor – e.g. a TMP36, a cheap single package device
- Relay, RC plug switches or a power MOSFET

- Screw terminals
- Box to trap the heat
- Heating element or incandescent bulb and fixture (or both)

The last item has been left deliberately vague. If you have an incandescent bulb (the kind that gets hot, not an energy-saving bulb), a low voltage lamp (like the old car headlights) or a hot lamp for sporting injuries and such, it's probably the easiest to set up. If you can procure a Peltier module[1] heating and cooling will be possible.

Requirements

1) The subsystem shall take commands (a command interface shall be defined) via a serial port connected to a computer (The USB serial is OK, you can use a terminal program for command and control).
2) It shall be possible to set temperature setpoints and the system shall command heating (or cooling) in the most power efficient way[2].
3) Safety mechanisms (current limitation and temperature limitation) shall be implemented.
4) The subsystem shall be FAILURE TOLERANT, a single failure in any of its components shall not compromise its operations.

2: Flexible Telecommand Decoder

What You Will Need

- Arduino/ARM development board
- Serial port input with a generator of telecommands using a standard format[3].

Requirements

The subsystem shall take a generic data stream via a serial port connected to a computer (The USB serial on Arduino UNO is NOT OK, use a dedicated port[4]).

1 Example of a suitable module: https://benselectronics.nl/peltier-element-60-watt/?utm_source=googleshopping&utm_medium=advertentie&utm_campaign=Feb2018/

2 Trade off using a PID control law, for example https://www.codeproject.com/Articles/36459/PID-process-control-a-Cruise-Control-example

3 You can use either the old PSS-04 standard format http://microelectronics.esa.int/vhdl/pss/PSS-04-107.pdf (check also decoder implementation requirements http://microelectronics.esa.int/vhdl/pss/PSS-04-151.pdf) or the more modern http://www.ecss.nl/wp-content/uploads/standards/ecss-e/ECSS-E-50-04A14November2007.pdf. Those documents are complex and might be confusing at a first read, for a simpler view start from MA28140 (PTD single chip) description http://microelectronics.esa.int/core/ipdoc/MA28140.pdf. In figure 3 you have to implement (at least) the coding layer.

The Packet Telecommand Decoder (PTD), shall hande (at least) 1 TC input channels, and process the following layers:

- Coding Layer (mandatory)
- Transfer Layer (optional)
- Segmentation Layer (optional)
- Authentication Layer (optional)

Some of these layers have a telemetry reporting mechanism (USB serial can be used for TM reporting).

The subsystem shall be **failure tolerant**, a single failure in any of its components shall not compromise its operations.

3: Intelligent sun sensor electronics

What You Will Need

- Arduino/ARM development board

In order to detect the intensity of light or darkness, you will use a sensor called a LDR (light dependent resistor[5]).

Requirements

The subsystem shall read LDRs placed on a representative microsat structure (a wooden/paper box) and derive the 'sun vector'[6]. The ATmega controllers used for the Arduino contain an onboard 6 channel (8 channels on the Mini and Nano, 16 on the Mega) analog-to-digital (A/D) converter. Consider this when deciding how many LDRs to use. You can also useIf you use the 4051 as a Multiplexer: You can choose between 8 different inputs and select just one you want to read at the time.
You should be able to evaluate the measurement error defined as pointing error of the sun vector measured versus the real one..
The subsystem shall be **failure tolerant**, a single failure in any of its components shall not compromise its operations.

4: Solar panel Drive Mechanism

Stepper motors, due to their unique design, can be controlled to a high degree of accuracy without any feedback mechanisms. The shaft of a stepper,

4 For a tutorial on ARDUINO serial look at https://www.arduino.cc/reference/en/language/functions/communication/serial/
5 This is an example https://benselectronics.nl/ldr-lichtsensor/?utm_source=googleshopping&utm_medium=advertentie&utm_campaign=Feb2018/ but there are many similar.
6 An halogen lamp can also work as directional light source.

mounted with a series of magnets, is controlled by a series of electromagnetic coils that are charged positively and negatively in a specific sequence, precisely moving it forward or backward in small "steps".

What you will need

- Arduino or ARM Board
- stepper motor[7]
- U2004 Darlington Array (if using a unipolar stepper)
- SN754410ne H-Bridge (if using a bipolar stepper)
- Power supply appropriate for your particular stepper

Requirements

subsystem should receive a command to set the solar panel at a certain angle, and implement it in a safe way, giving feedback.
The subsystem shall be **failure tolerant,** a single failure in any of its components shall not compromise its operations.

Timeline and effort distribution

Week	**3.8**	**3.9**	**4.0**	**4.1**	**Total**
Dates	3-6 April	9-13 April	16-20 April	23-26 April	
Effort (hrs)	8	8	8	8	32

Week	**Activity**
3.8	Project plan (no Gantt chart required), engineering approach, requirements definition
3.9	Design of the subssystem and failure mode analysis
4.0	Implementation in hardware & software
4.1	Verification of the design and writing the final report

7For this a simple analog feedback servo (that does not need interface electronics) can do the job. Please look at https://www.kiwi-electronics.nl/componenten-onderdelen/motors-servos/analog-feedback-micro-servo-metal-gear as example, but do a tradeoff for the final choice (also optical feedback motor are available, as full stepper).

TU Delft	**AE4265** **Group Assignment Description**

Group size
The assignment is expected to be carried out by **2** students.

Deliverable Items

The final deliverables are: final report and prototype (hardware & software).

Grading criteria

The group assignment final deliverables will be graded based on the following criteria:

- Understanding of the requirements
- Engineering approach
- Functionality and usability of the deliverable
- Planning
- Creativity
- Independence

References

1. Lectures' slides
2. Peter Marwedel "Embedded System Design: Embedded systems, Foundations of Cyber-Physical Systems and Internet of Things", 3rd edition, 2018, Springer
3. JPL coding standards (public wersion, without MISRA C) https://lars-lab.jpl.nasa.gov/JPL_Coding_Standard_C.pdf
4. An example of typical coding faults in embedded systems https://embeddedgurus.com/barr-code/2011/03/unintended-acceleration-and-other-embedded-software-bugs/
5. http://sunnyday.mit.edu/caib/concepts.pdf This white paper lays out some foundational information about different approaches to safety: how various industries differ in their approaches to safety engineering, and a comparison of three general approaches to safety (system safety, industrial safety engineering, and reliability engineering). An attempt is made to lay out the properties of industries and systems that make one approach more appropriate than another.