Utilizing Low-Cost Linux Micro-Computer & Android Phone Solutions on Cube-Satellites

Ahmed Farid[1], Ahmed Samy[2], Ahmed Shalaby[2], Ahmed Tarek[2], Mahmoud Ayyad[2], Muhammad Assem[2], Samy Amin[2]

[1] B.Sc. Graduation project, Computer Engineering Department, October University for Modern Sciences & Arts

[2] B.Sc. Graduation project, Aerospace Engineering Department, Cairo University

Outline

- Problem Definition
- Proposed Approach
- Approach Description
- Linux Micro-Computers
- Android Smartphones
- Interface Models
- Proposed Improvements
- Conclusion

Problem Definition

- Implementation of satellite computing functionalities as a proof of concept, given the following circumstances:
  - Minimal time span (1 Year + No prior space know-how).
  - Lesser funding for overall project. (Student-funded)
  - Lack of/Import restrictions on certain components.
- The following capabilities are required for the onboard computer:
  - Autonomous operation.
  - Operating a payload camera
  - Providing necessary interfaces. (i.e. I2C, SPI, A/D… etc.)
  - Real-time system monitoring.
  - Bidirectional communications handling.

Proposed Approach

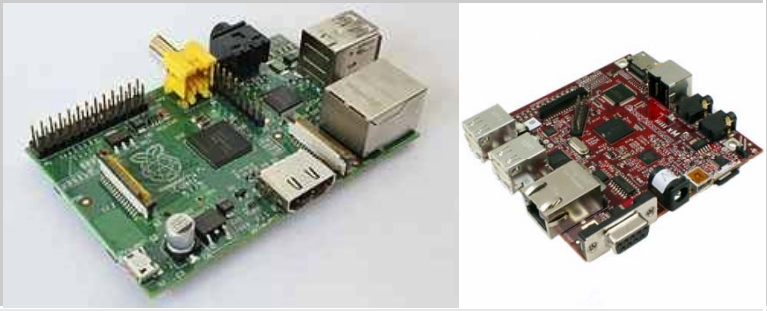
Recommended Strategies:

- Budget cannot afford space-qualified systems.
- More focus on local markets and affordable options.
- Design Complexity = Possible issues = More troubleshooting time.

Suitable approaches:

- Linux Micro-Computers
- Android Smartphones

Approach Description

	Linux Micro-Computers	Android Smartphone
Picture		
Specifications	- Low cost (25-200$) -700-1000 MHz, 512 Mb RAM - SD Card, USB, Ethernet, Audio - Hardware I/O (µART, SPI, I2C, GPIO) - Power: 1-2.3 W - Weight: 37-45 gm	- Relatively low-cost (300-500$) -Dual 1 GHz processor - 512-2 Gb RAM - Integrated camera, sensors, and SD card - Power: Device and usage dependant (Ex: Galaxy S3 is 0.45 W on normal use.) - Weight: Device dependant
Disadvantages	- Missing hardware IO in some OBCs (Analog, PWM as in the **Raspberry Pi**)	- No built-in hardware I/O

Linux Micro-Computers:
Sensor Readings Acquisition & Actuator Control

Requirements

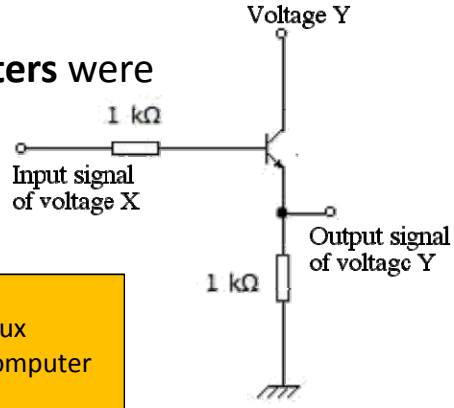
- Interface with sensors (I2C, SPI, A/D)
- Control actuators (PWM)

Complications

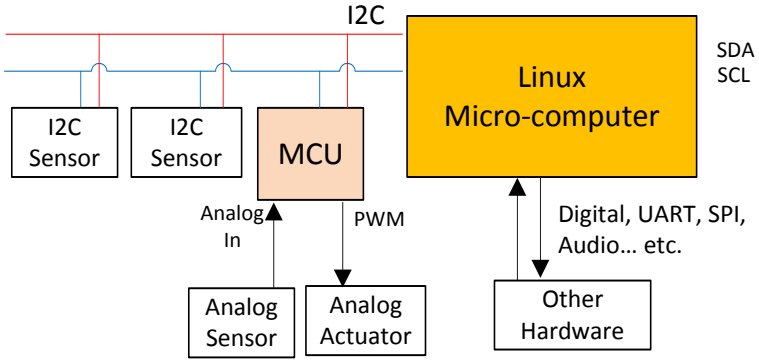
- Some interfaces may not be available on accessible OBCs (PWM, A/D)
- Difference in voltage logic levels

Implementation

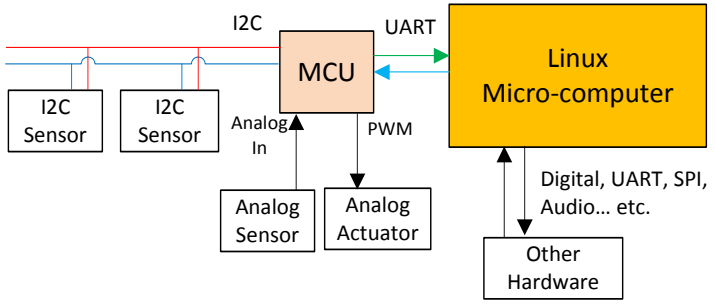
- An **external MCU** is used, acting as a compatibility layer (Ex: PIC16f877). Connections are over I2C or µART channels.
- Simple **voltage level shifters** were implemented (5-3.3 v).



Transistor-based Logic-Level Shifter



I2C configuration



Serial configuration

Linux Micro-Computers:
Operating Payload Camera

Requirements

- Camera interface for image capturing
- Image storage

Complications

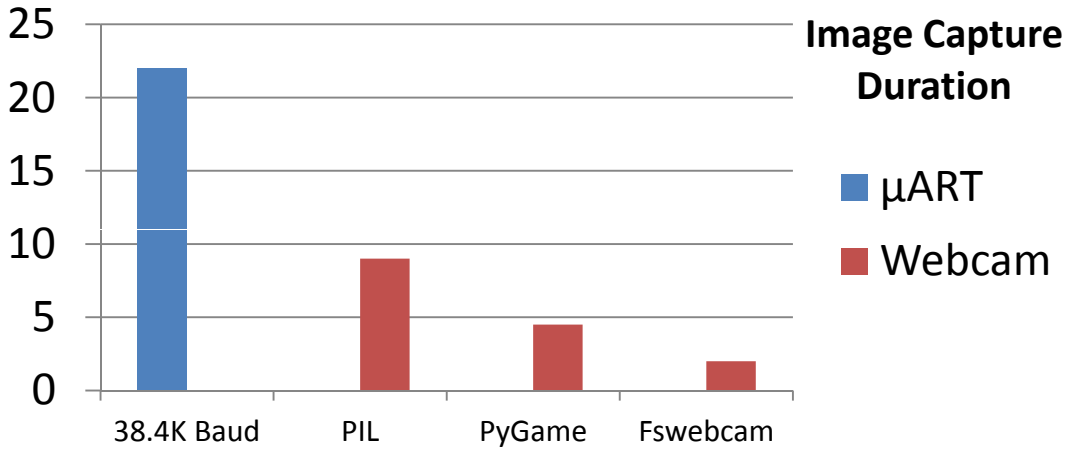
- UART and I2C cameras in local markets are slow in storage process.

Implementation

- Usage of **USB cameras**. (Typically webcams)
- Since Linux is used, interface can be done using **programming libraries** or **ready-made packages** (fswebcam).
- Capture takes **2 seconds** for 320x240 images.

Linksprite µART Camera

Playstation Eye Webcam



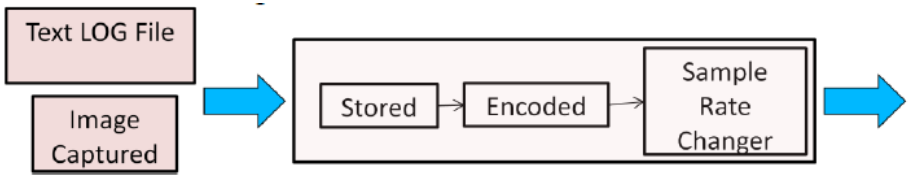
Linux Micro-Computers: Communications

Requirements

- Send/Receive Data with a ground-station
- Interpret commands from a ground-station

Complications

- Due to import restrictions, no low-level long range VHF/UHF transceivers are sold.
- Only hand-held radio transceivers (VHF/UHF) are available, with audio interface only.



Data encoding workflow

Implementation

- **I2C**, **SPI**, or **UART** are utilized in the presence of low-level communications hardware.
- **Robot36** was used to convert images to 36-second sound. Modifying sampling rate helps reduce transmission time.
- **AFSK** was used to convert binaries to sound at 1200 bps transmission speed.
- **DTMF** was used to create downlink frame headers-trailers, and as uplink commands

Downlink Transmission Example

Beacon | Type: Binaries | Binaries | Type: Image | Images | END

Linux Micro-Computers:
Computation & Operations Management

Requirements

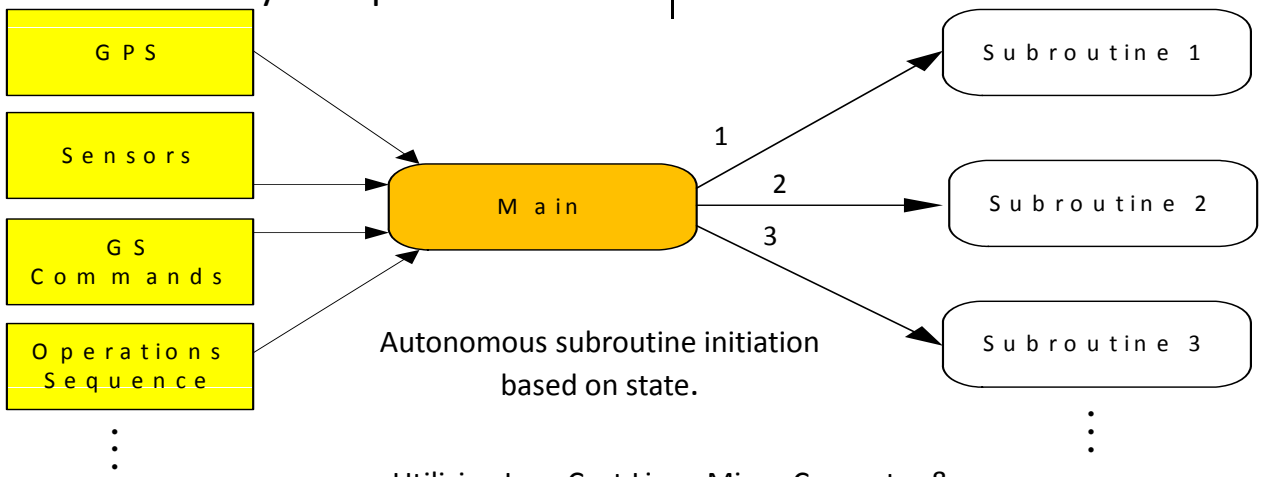
- Management of subroutines and modes.
- Use of an attitude determination algorithm.

Complications

- Prototypes and libraries not in C are time consuming in code porting.
- Integrating several modules in a single C program is considerably complex.

Implementation

- Modules are compiled as standalone, but called by a single controller program.
- The controller program has two main parts: Listener and state computation.
- Attitude determination written in *Matlab* script is run using Octave.



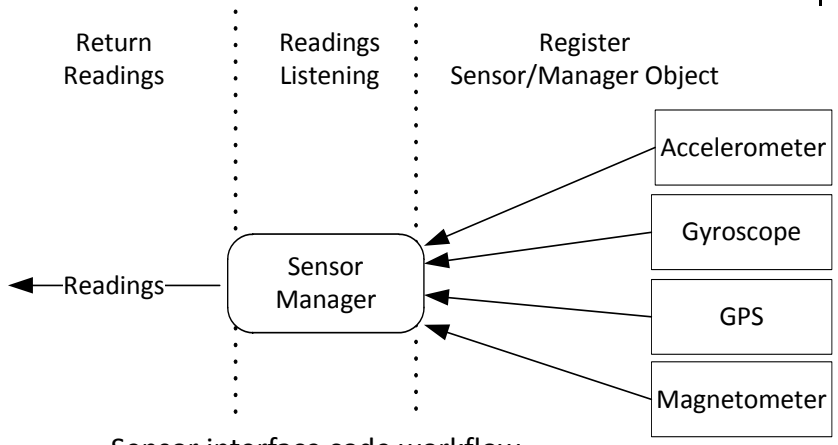
Android Smartphones:
Sensor Readings Acquisition, Actuator Control, & Image Capture

Requirements

- Interface with sensors .
- Control actuator
- Image capture and storage.

Complications

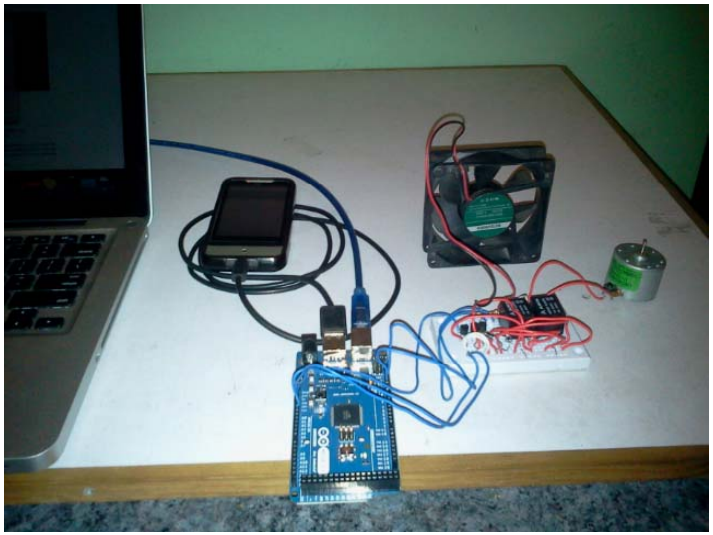
- Phone has no low-level hardware interface.



Sensor interface code workflow

Implementation

- Sensors and camera come **built-in**. Only **Java coding** is required.
- **OTG connection to external MCU** (Ex: IOIO, Arduino, Attiny85) for actuators.



Android app controlling DC fan and motor with MCU

Android Smartphones:
Computations & Operations Management

Requirements

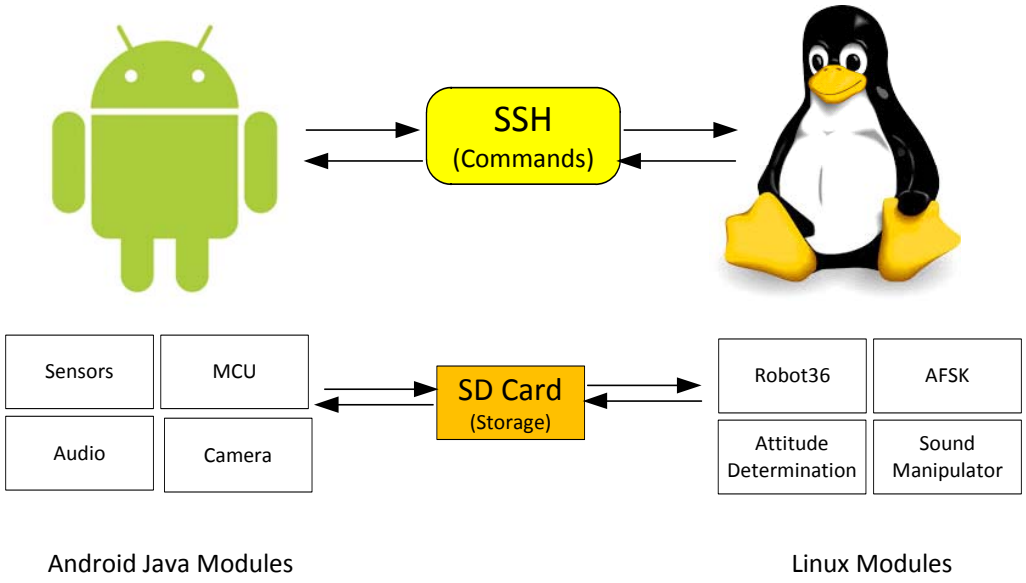
- Management of subroutines and modes.

Complications

- Prototypes and libraries not in Java/C are time consuming in code porting.

Implementation

- Virtual Linux OS is run in parallel to the native Android OS. (Initiated through CHROOT)



Android Smartphones:
Communications & Attitude Determination

Requirements

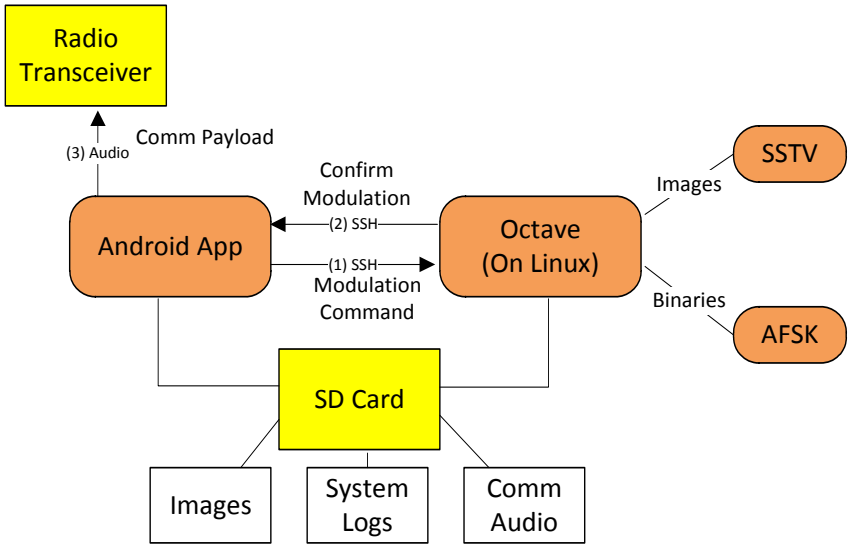
- Manage communications with the ground.
- Use of an attitude determination algorithm.

Complications

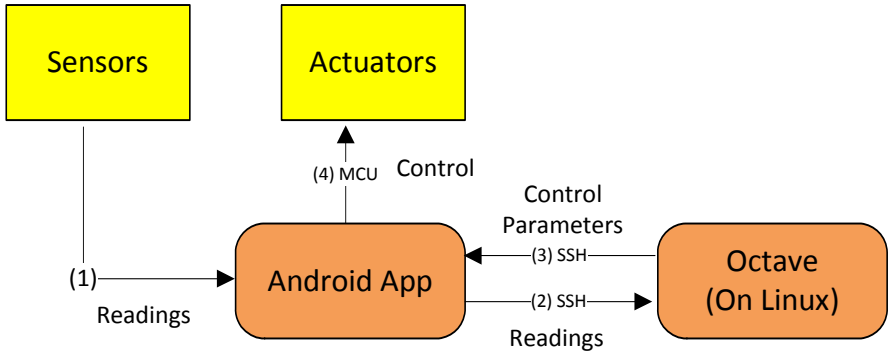
- Porting modules to Android is time costly.

Implementation

- Using **virtual Linux technique**: Codes, libraries and software packages implemented on Linux micro-computers are easily ported to Android (With minor modifications).
- **Robot36**, **AFSK**, and **Octave** are utilized.

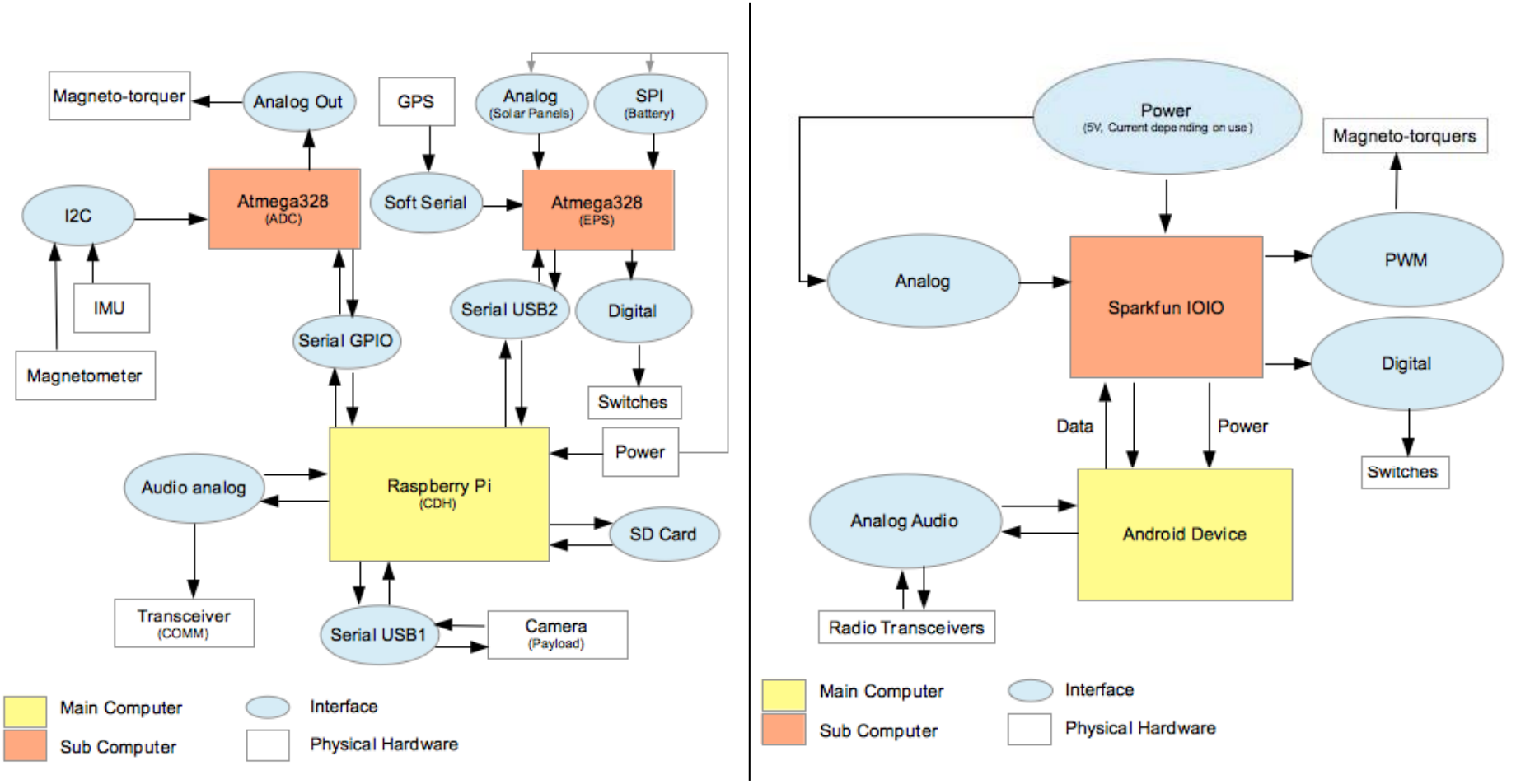


Android communications workflow

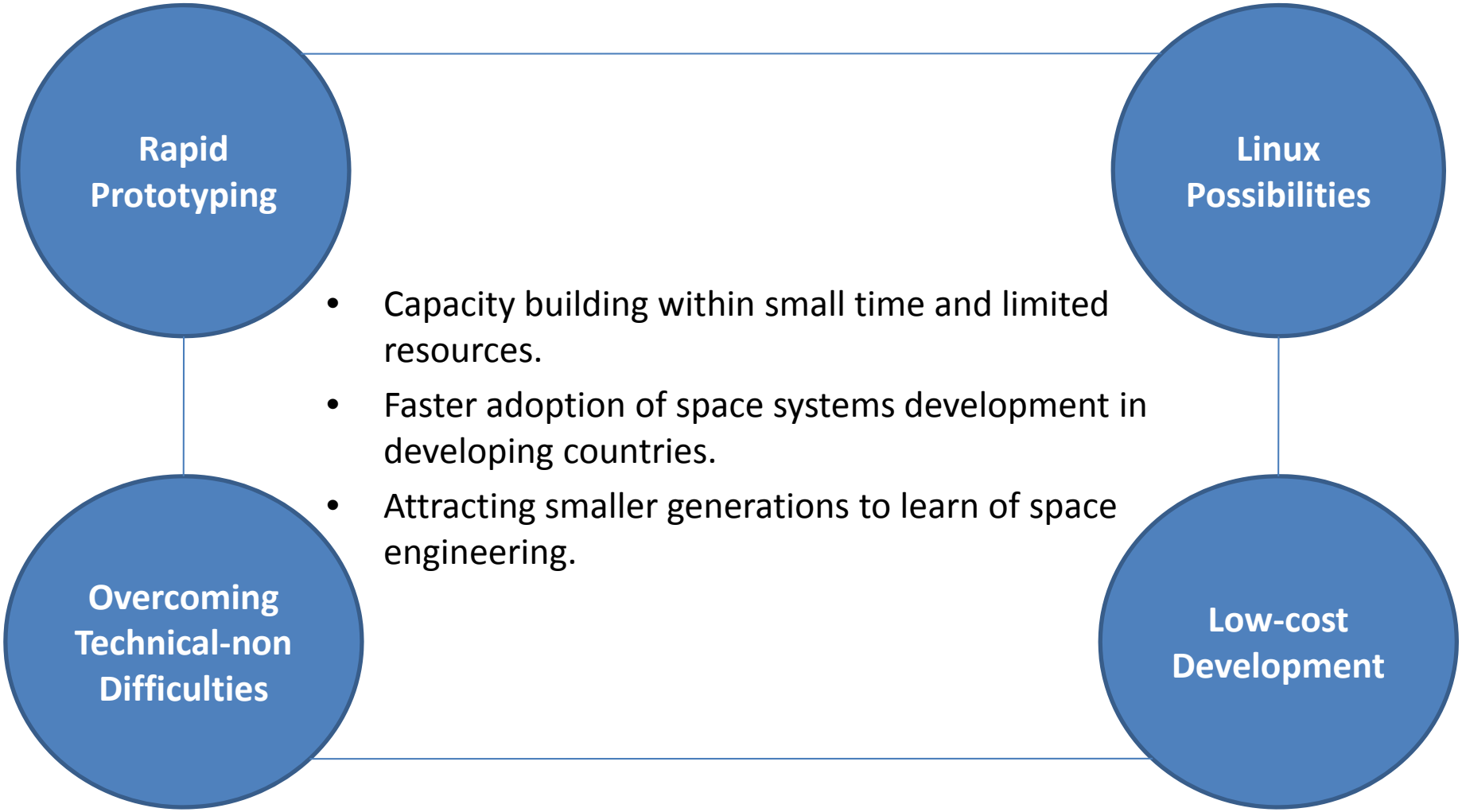


Android ADC workflow

Interface Models

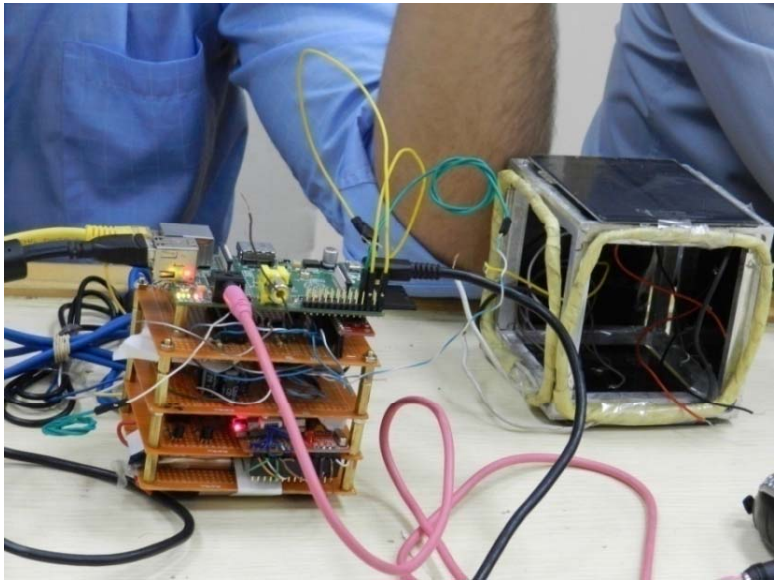


Conclusion



- Capacity building within small time and limited resources.
- Faster adoption of space systems development in developing countries.
- Attracting smaller generations to learn of space engineering.

Thank you for your time.



Acknowledgements

- Our mentors: ***A. Sherif***, ***A. Desoki***, and ***M. Khalil*** for their highly valued assessments and supporting us with their knowledge and expertise in the field.
- **My project colleagues** for their cooperative work to reach project milestones.
- Our **families and friends** for their continued moral support.

The overall project is presented during the poster session as #32.