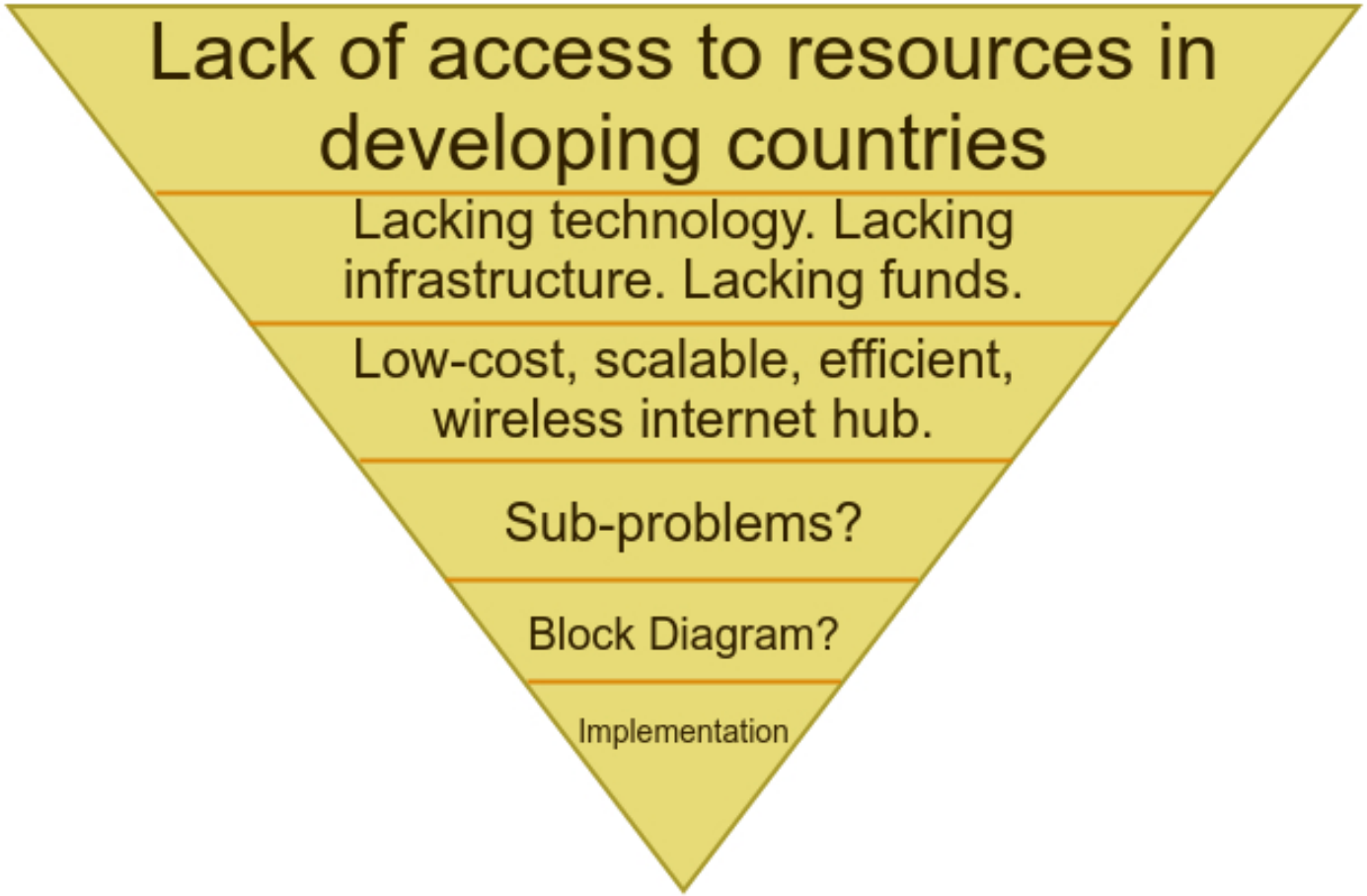
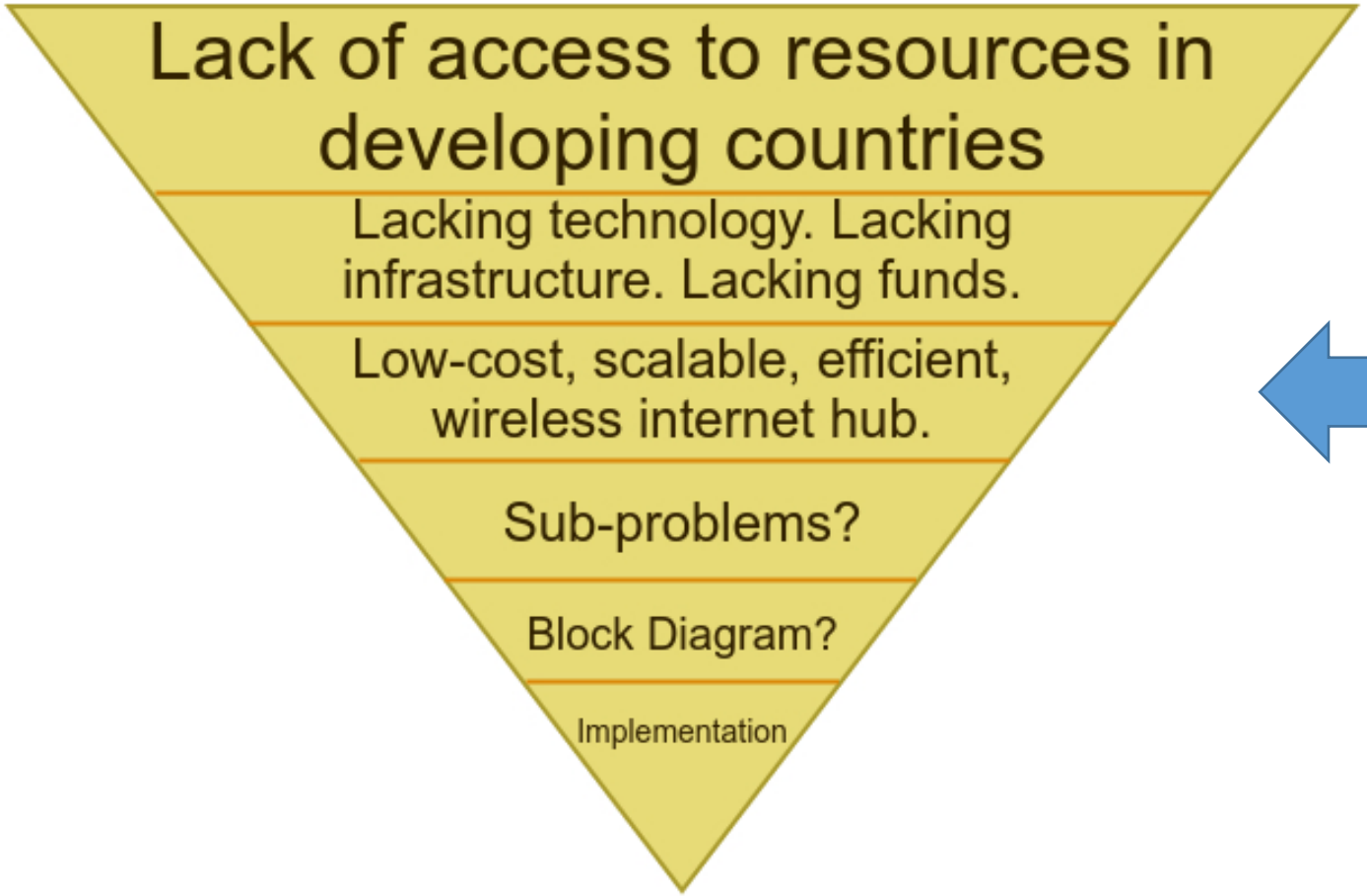
Lecture 6
Requirements and Verifications

Engineering (or Requirements) Flow-down

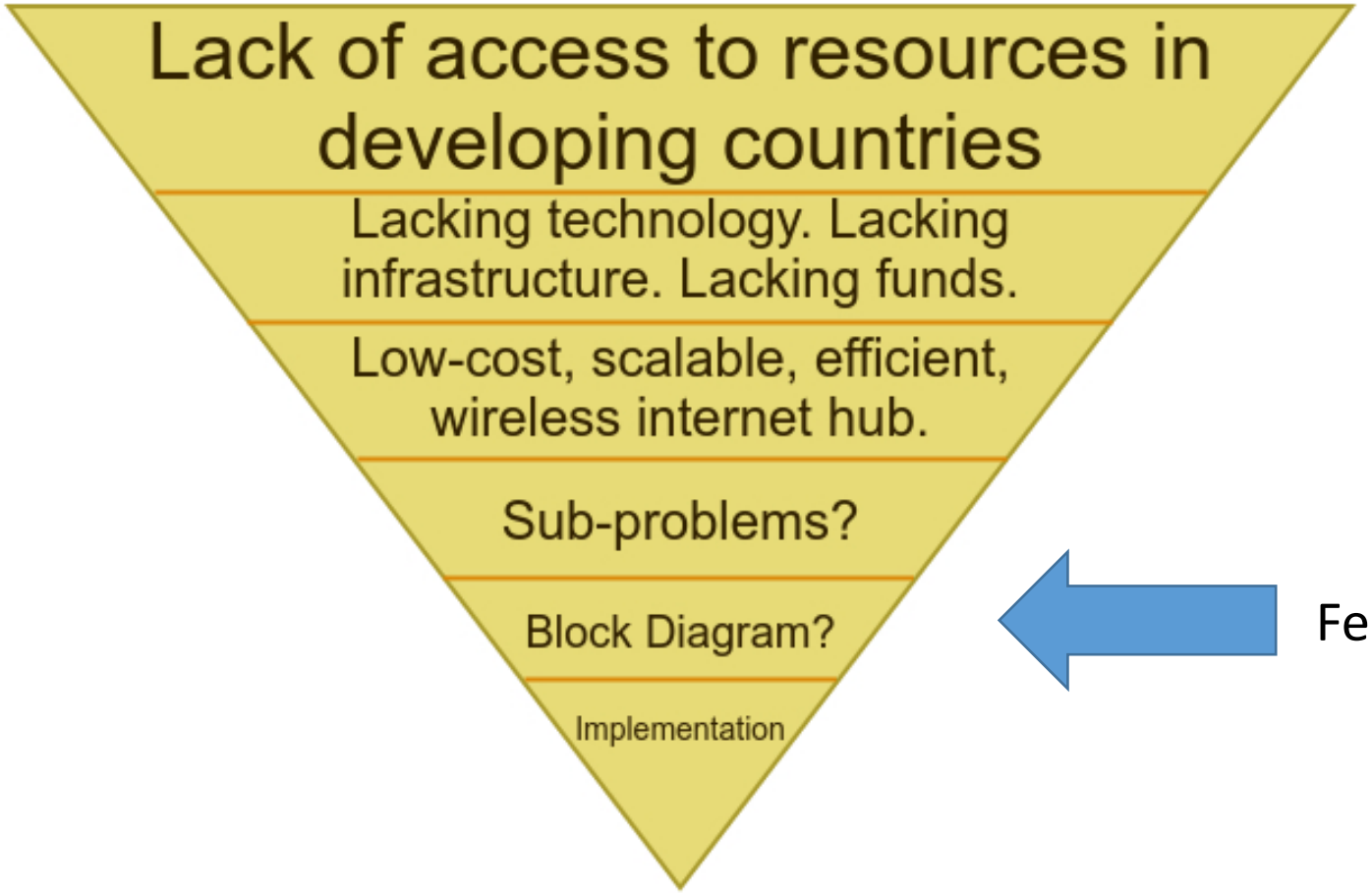


Engineering (or Requirements) Flow-down



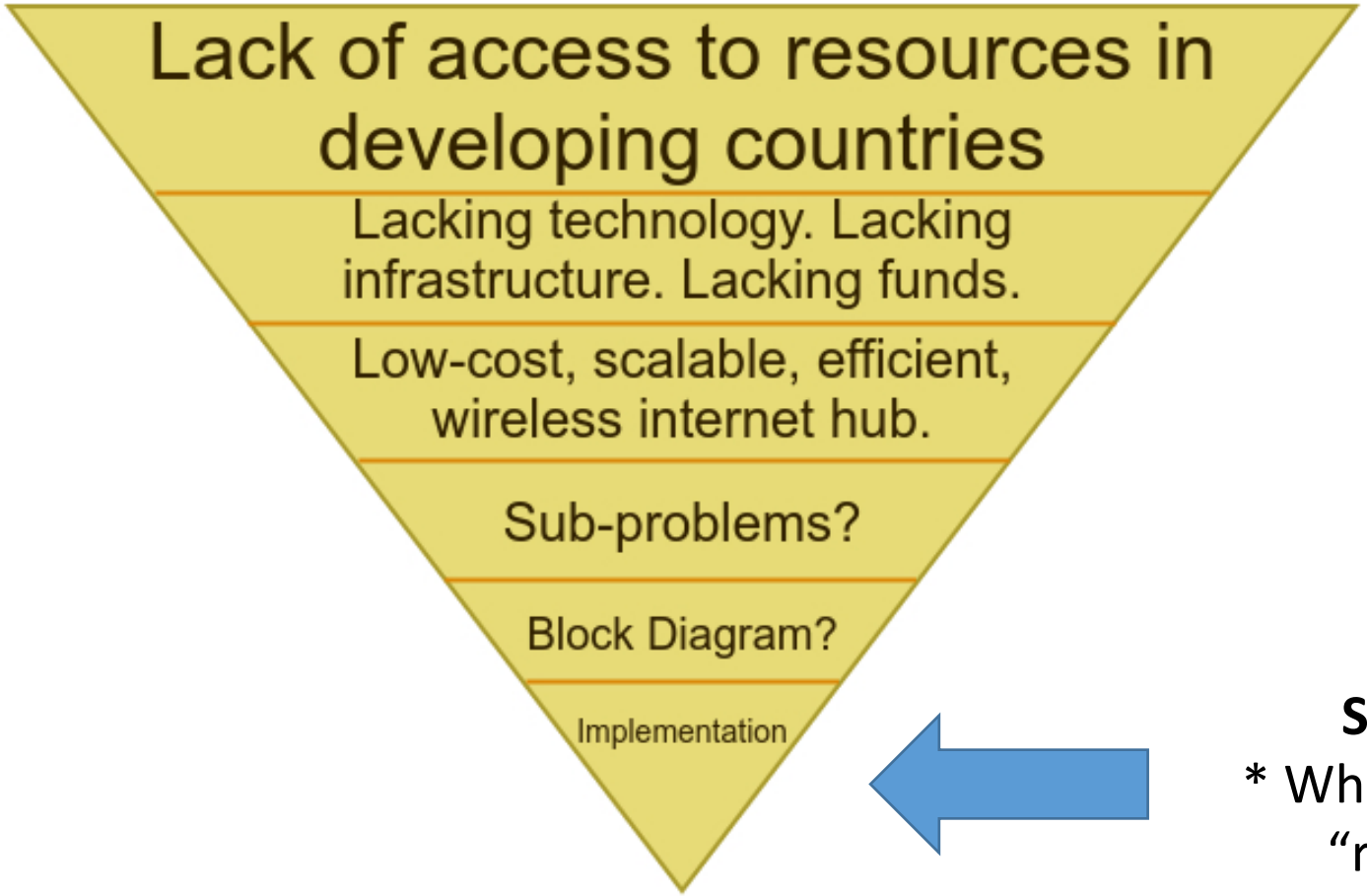
High-level or “customer” requirements.

Engineering (or Requirements) Flow-down



Feature-level design

Engineering (or Requirements) Flow-down



Specifications:
* What 445 refers to as "requirements".

High Level Requirements of a Block Diagram

- **Modularity**
  - Independent
  - Separate Functions
  - Clear Boundaries

- **Information Flow**
  - Clear inputs and outputs
  - Clear flow of information
  - Clearly shows what the information is

- **Justified Design**
  - Functionality is clear (high-level and module functions)
  - High-level reqs → Feature reqs → specifications
  - Design choices are outlined and defensible

High Level Requirements of a Block Diagram

- **Modularity**
  - Independent
  - Separate Functions
  - Clear Boundaries

"Each block should be as modular as possible. In other words, they can be implemented independently and re-assembled later."

Organized hierarchically. The entire system is composed of sub-systems, these subsystems contain low level blocks.

We will see an example shortly.

High Level Requirements of a Block Diagram

- **Information Flow**
  - Clear flow of information
  - Clear inputs and outputs
  - Clearly shows what the information is

Diagrams should include connectors that inform the reader of how information, power, etc. are transferred between blocks in the diagram.

This should include directionality of flow and a clear graphical notation of what the flow represents.

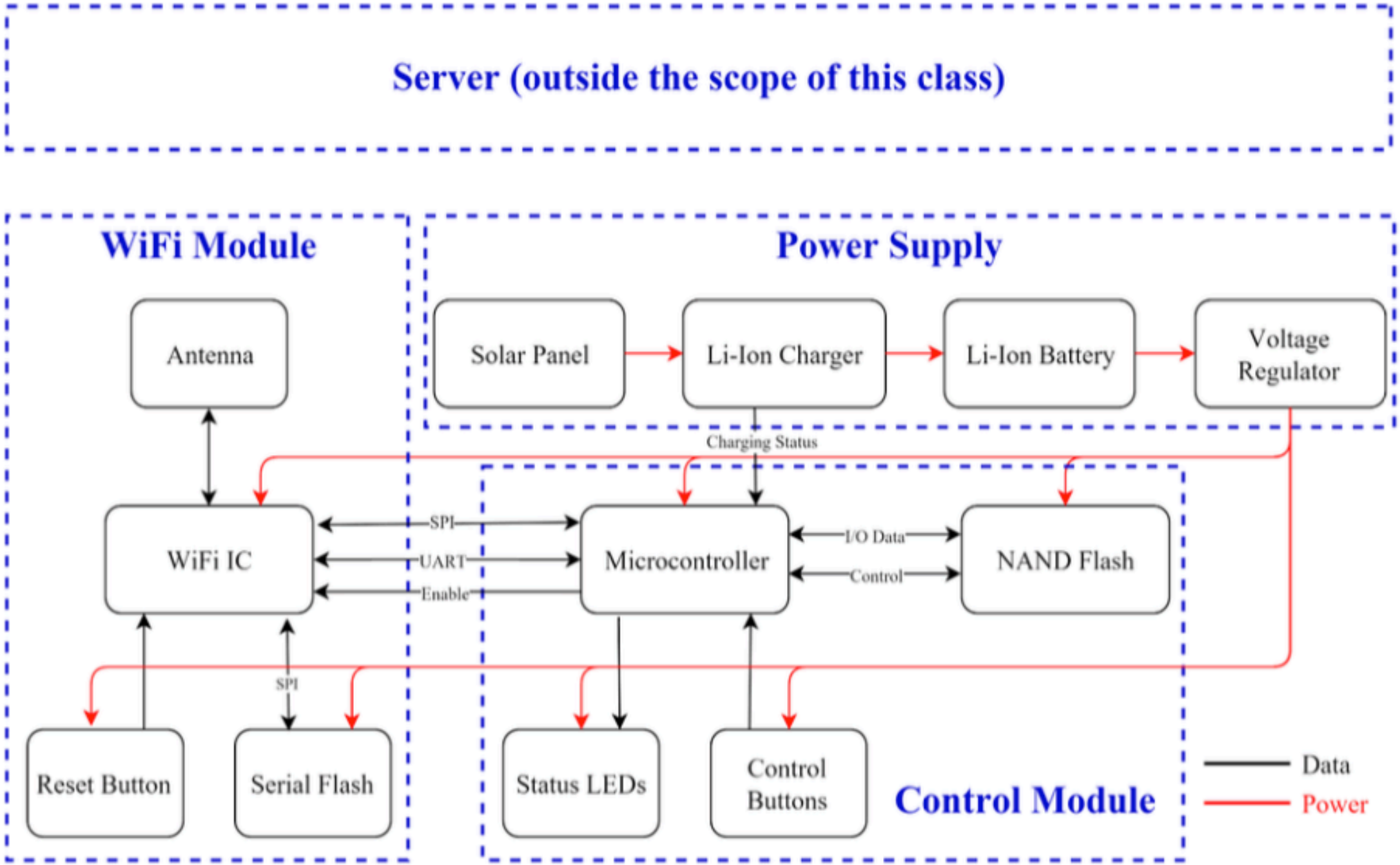
High Level Requirements of a Block Diagram

- **Justified Design**
  - Functionality is clear (high-level and module functions)
  - High-level reqs → Feature reqs → specifications
  - Design choices are outlined and defensible

Each block in your block diagram will be accompanied by one paragraph of text. These paragraphs are boring, they should take a standard form, for example:

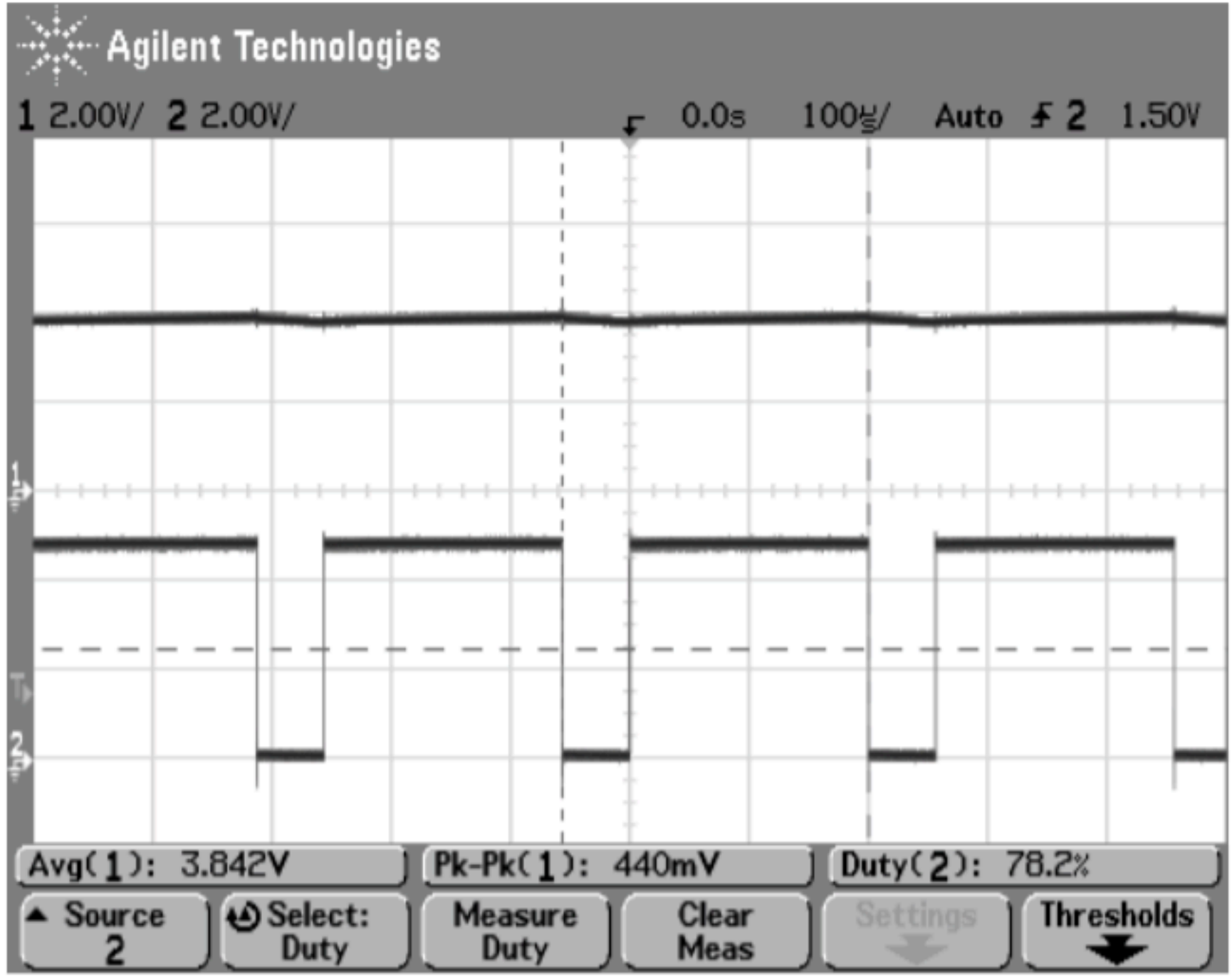
Name of block.

- Summary sentence, purpose of block.
  - This block receives inputs from where?
  - This block does what to these inputs?
  - This block was chosen (as opposed to other solutions), because…
  - This block sends outputs to where?


Server (outside the scope of this class)
WiFi Module
Power Supply
Antenna
Solar Panel
Li-Ion Charger
Li-Ion Battery
Voltage Regulator
Charging Status
WiFi IC
SPI
UART
Enable
Microcontroller
I/O Data
Control
NAND Flash
SPI
Reset Button
Serial Flash
Status LEDs
Control Buttons
Control Module
Data
Power

2.2.1 Microcontroller (Application Processor)

The microcontroller, a Freescale Kinetis KV30P64M100, handles memory allocation for the cache. It communicates with the WiFi chip via UART, and reads the NAND flash cache through SPI. This microcontroller was chosen for its affordability and SPI clock speeds of 12.5MHz, which forms the basis for the WiFi module data transfer so users can access data as quickly as possible. The chip also monitors battery charge status and will disable the WiFi module and itself if the battery voltage is low enough to risk damaging the battery. The microcontroller is programmed through a JTAG interface by a Freescale Freedom development board.


Agilent Technologies
1 2.00V/ 2 2.00V/
0.0s
100µs/
Auto
2 1.50V
Avg(1): 3.842V
Pk-Pk(1): 440mV
Duty(2): 78.2%
Source 2
Select: Duty
Measure Duty
Clear Meas
Settings
Thresholds

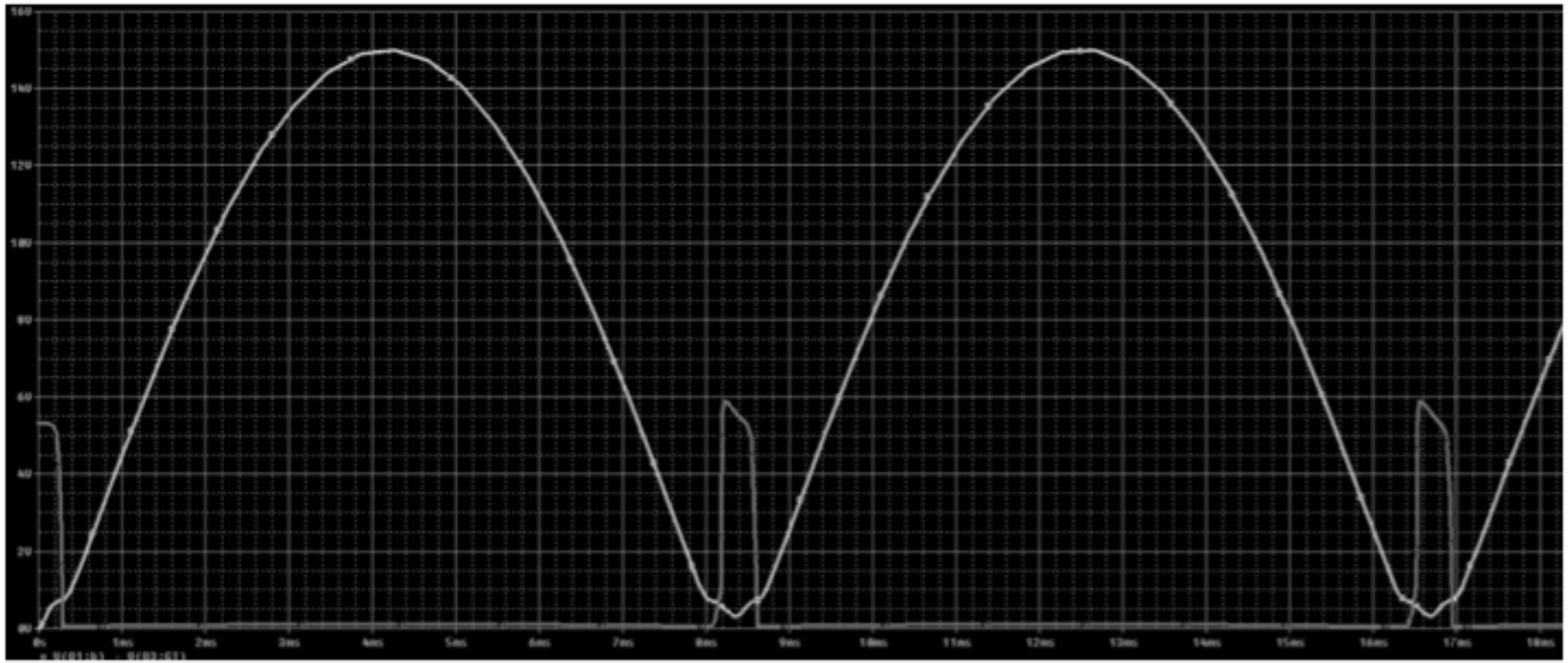
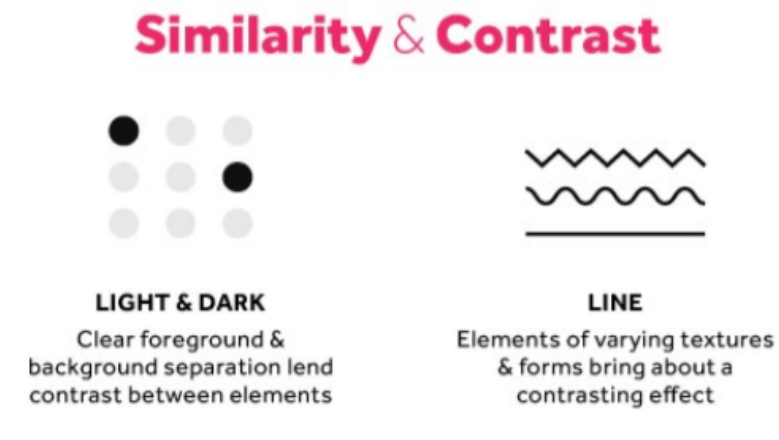
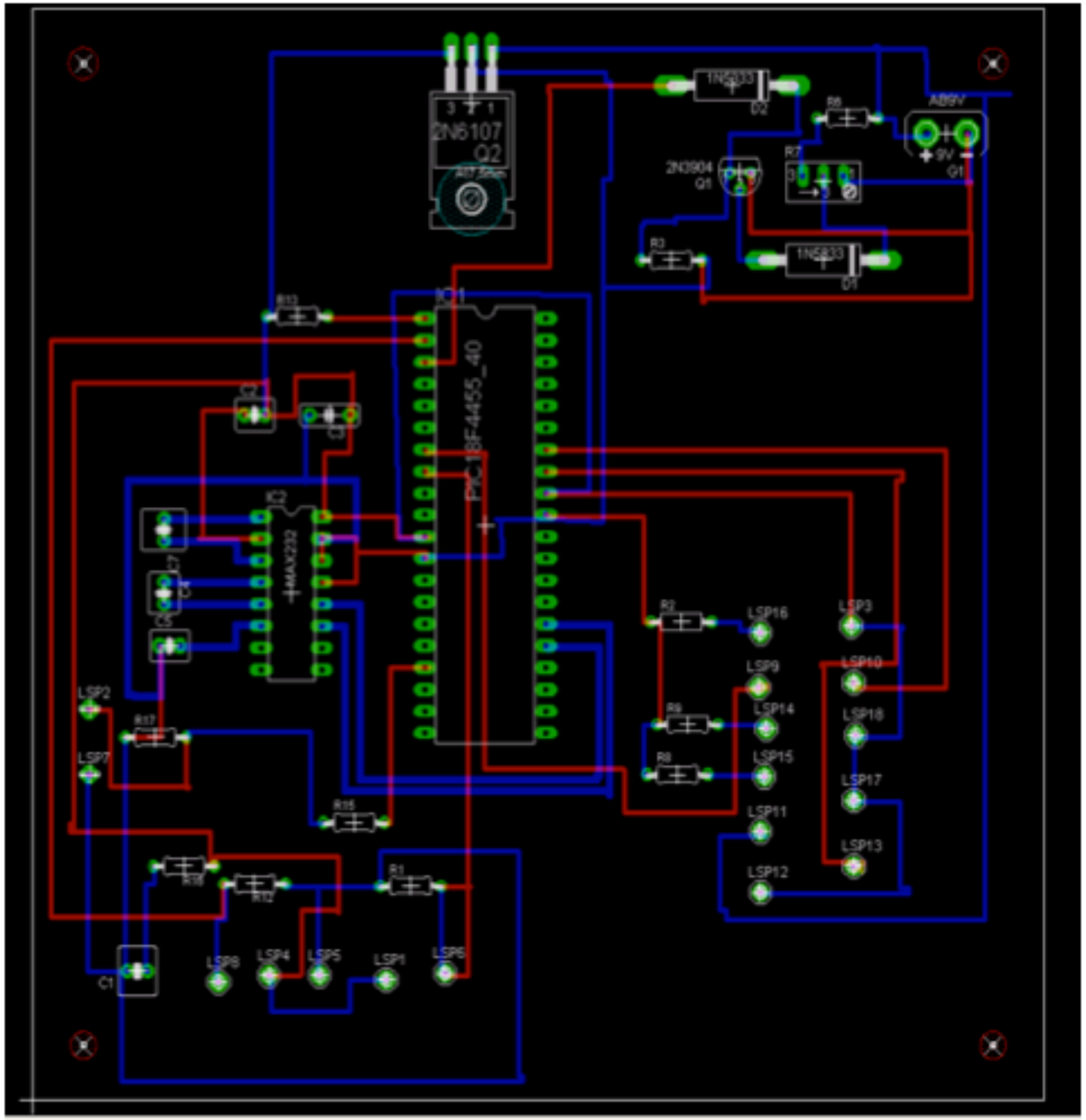
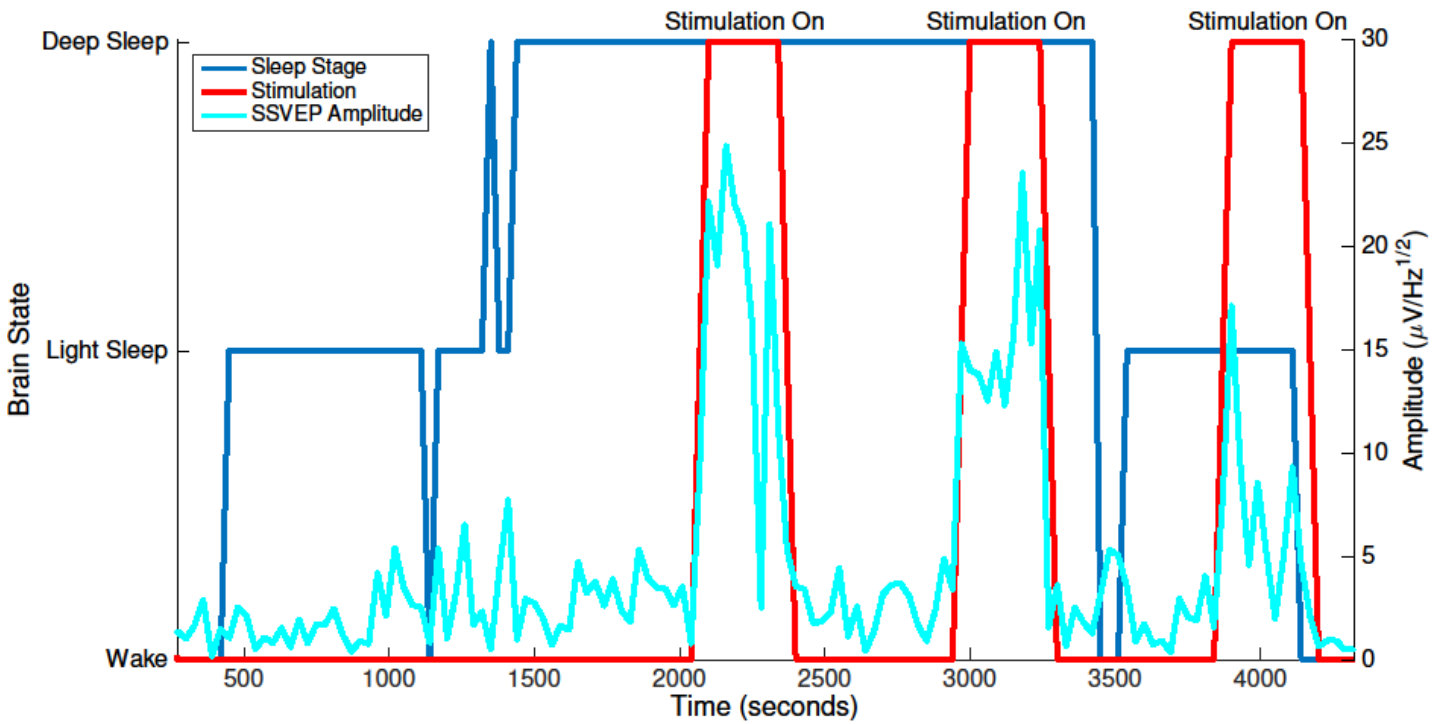


Figure A. 1: Rectified sine wave and BJT output





2N6107
Q2
1N5833
D2
R6
AB9V
9V
2N3904
Q1
R7
R3
1N5833
D1
IC1
R13
C2
C3
IC2
MAX232
C4
C5
LSP2
R17
LSP7
R15
R18
R12
R1
C1
LSP8
LSP4
LSP5
LSP1
LSP6
R2
LSP16
LSP3
LSP9
LSP10
R9
LSP14
LSP18
R8
LSP15
LSP17
LSP11
LSP13
LSP12



Dominance / Emphasis

HIGHLIGHT

Breaking the visual hierarchy using form to lay emphasis

COLOUR

To distinguish between elements in a series of similar forms

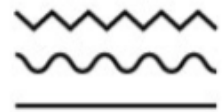
SIZE

Elements of different sizes focus the viewers attention accordingly

Similarity & Contrast

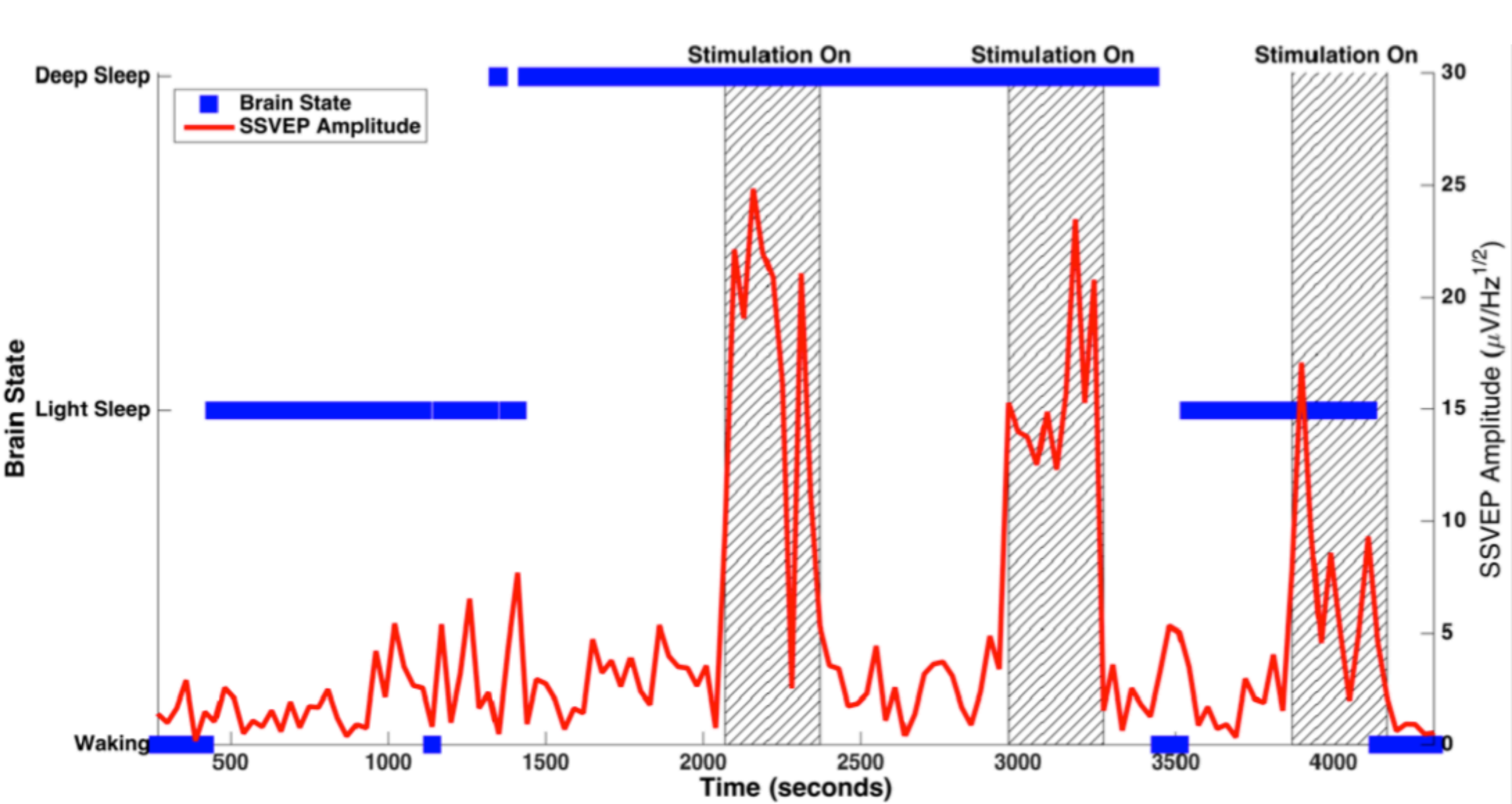
LIGHT & DARK

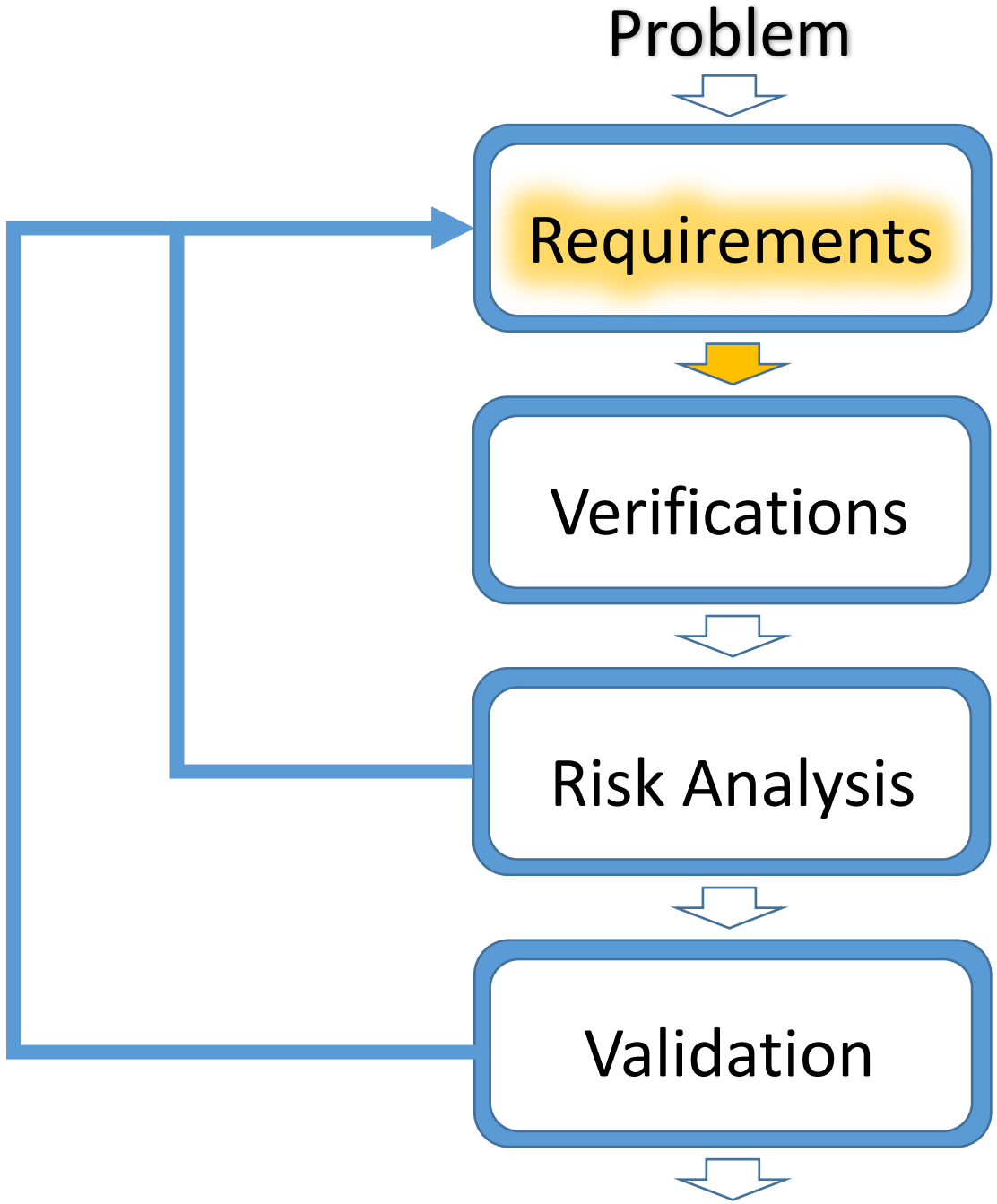
Clear foreground & background separation lend contrast between elements



LINE

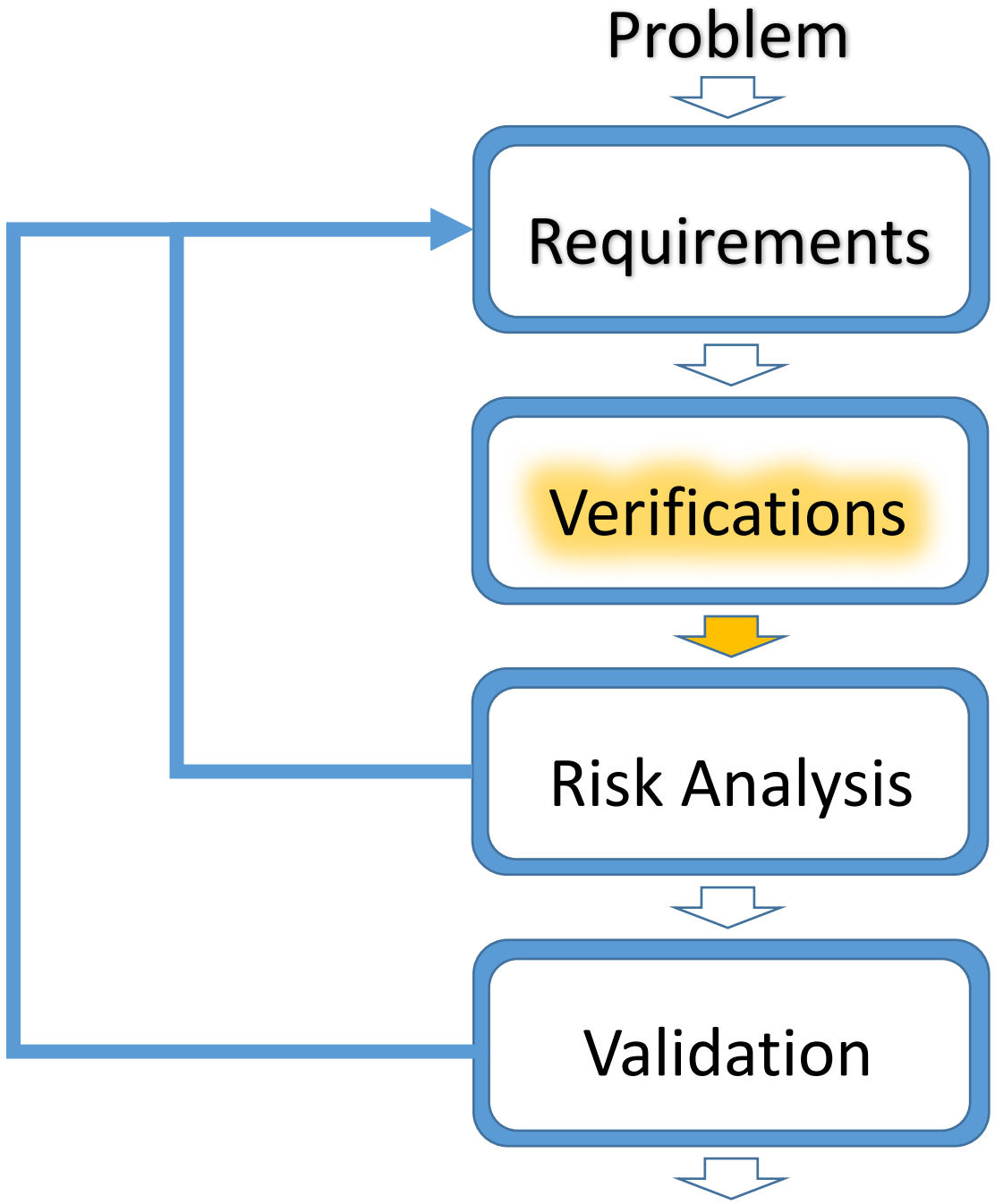
Elements of varying textures & forms bring about a contrasting effect


Stimulation On
Stimulation On
Stimulation On
Deep Sleep
Light Sleep
Waking
Brain State
Brain State
SSVEP Amplitude
Brain State
SSVEP Amplitude (μV/Hz^1/2)
30
25
20
15
10
5
0
500
1000
1500
2000
2500
3000
3500
4000
Time (seconds)



Requirements: (specifications)

A set of statements that describe the attributes your system must have to solve your problem.



Verifications:

A set of statements that describe tests you will perform to make sure that your system meets the requirements.

An introduction to specifications

- Berkun describes a story in “getting things done”.
  - Given an incredibly long template by boss.
  - Programmers laughed at it, said specs weren’t needed.
  - Would you say the same to a civil engineer?
  - Not all projects are the same.
    - People’s lives rarely depend on software?
  - Most projects, however:
    1. Are done by a team
    2. Have budgets and deadlines
    3. Need a way of formalizing knowledge of the project
  - Solution was to go back to boss and rework the specification template.

The point…

- Surprise surprise, there are many ways to do specifications.

The value of specifications

- 3 different perspectives
  - Team
  - Project
  - Stakeholder

From the team's perspective

- A good set of specifications should:
  - Solve problems for the team
  - Accelerate work
  - Increase the probability of producing a quality project
  - Should meet the needs of the team and the project
- As with everything we talk about, they are arrived at through iteration.

From Berkun, "Making things happen"

From the project's perspective, specifications:

1. Ensure the right things are built
   - This is what we are discussing today
2. Provide a schedule and milestones
   - We will talk schedules in a few weeks
3. Enable deep review and feedback
   - We will have a formal design review at the end of the semester

From Berkun, "Making things happen"

From a stakeholder perspective

- Used correctly, convey information in a simple and easy to understand way to team members and interest holders.

- “ When used poorly, they are hard (or painful) to read, tedious to create, and frustrating for everyone who comes in to contact with them.” - Berkun

- A lack of interest in specs is often related to a failed understanding of what they are, what they can do for you, and how to write good ones.
  - I see this often…..especially difficult in technical fields.

From Berkun, “Making things happen”

What can specs do for you?

- Describe the functionality of what will be built
- Help designers clarify decisions **by forcing them to be specific**
  - I have seen this many times, where a well done job enables criticism…
- Allow people to question/give feedback
- Communicate from one to many
- Create a team-wide point of reference and a living description of the design
- Provide insurance for loss of team members
- Add sanity to the team and a record of what was promised.

From Berkun, “Making things happen”

What specs cannot do for you?

- Eliminate discussions between team members
- Prove that the authors/creators are smart
- Prove how important a feature is
- Convert people to a certain point of view
- Show how great someone is at Visio

From Berkun, “Making things happen”

Specifying is not designing!

- These are two different processes.
- Designing involves creativity and choice, specifying is the act of expressing the details of an existing plan.
- Specifying should focus on explaining.
- Specifying something should be written in as clear a manner as possible, the way you arrived at a specification may not be the best way to describe that specification.

From Berkun, “Making things happen”

What does a good specification look like?

- Complicated does not equal good. The goal of a specification “is to describe things in a way that minimizes the amount of work other people have to do to understand it.”
  - This is, of course, a matter of judgement…what is good and what is complex are subjective.
  - Sometimes, however, complexity is a cover-up for a lack of understanding.

From Berkun, “Making things happen”

Tips for writing good specifications

- Pillage! Look at old specifications, reuse them, rewrite them, credit them, and move on.
- Avoid jargon…this is a theme.
- Is this a reference or a specification?
  - You don’t need to describe every concept (e.g. – voltage)
- Sketch things out
  - Pseudocode for coding specs
  - Table for hardware specifications

From Berkun, “Making things happen”

When are specs complete?

- Good question…

Define with reference to 398

Requirements of requirements (specifications)

- Quantifiable
  - Should involve a number. Remember our discussions on reducing ambiguity? Words like “very”, “many”, “lots”.
- Relevant
  - A specification for a DC motor probably shouldn’t include that it must be fuzzy or purple…unless it should.
- Detailed
  - A specification for a part should provide enough detail that anyone can read it and understand it.

Requirements of verifications

- Include measurement.
- Procedure for conducting measurement.
- Evidence that will be provided in report that requirement has been met.

What do you write specifications for?

- The easy rule is that every block from your block diagram will have a SET of specifications associated with it.
- Each specification will have a verification.

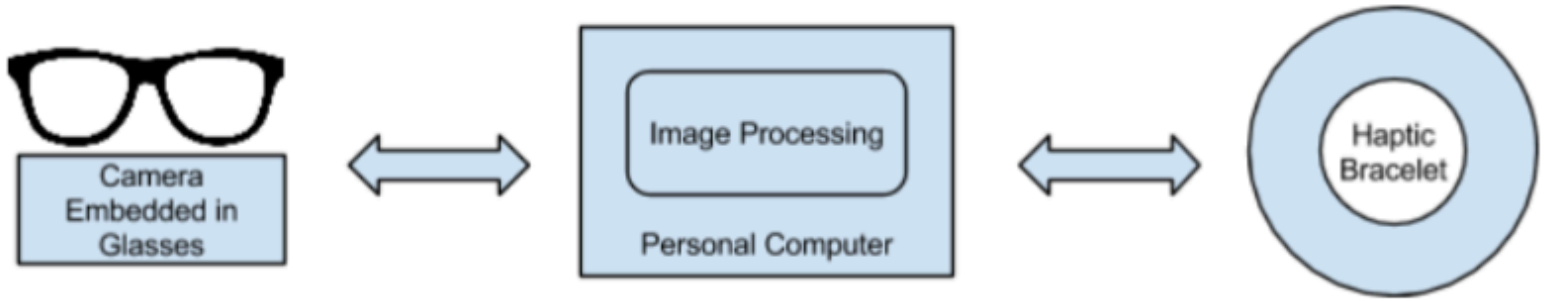


Figure 1: High level overview of our modules

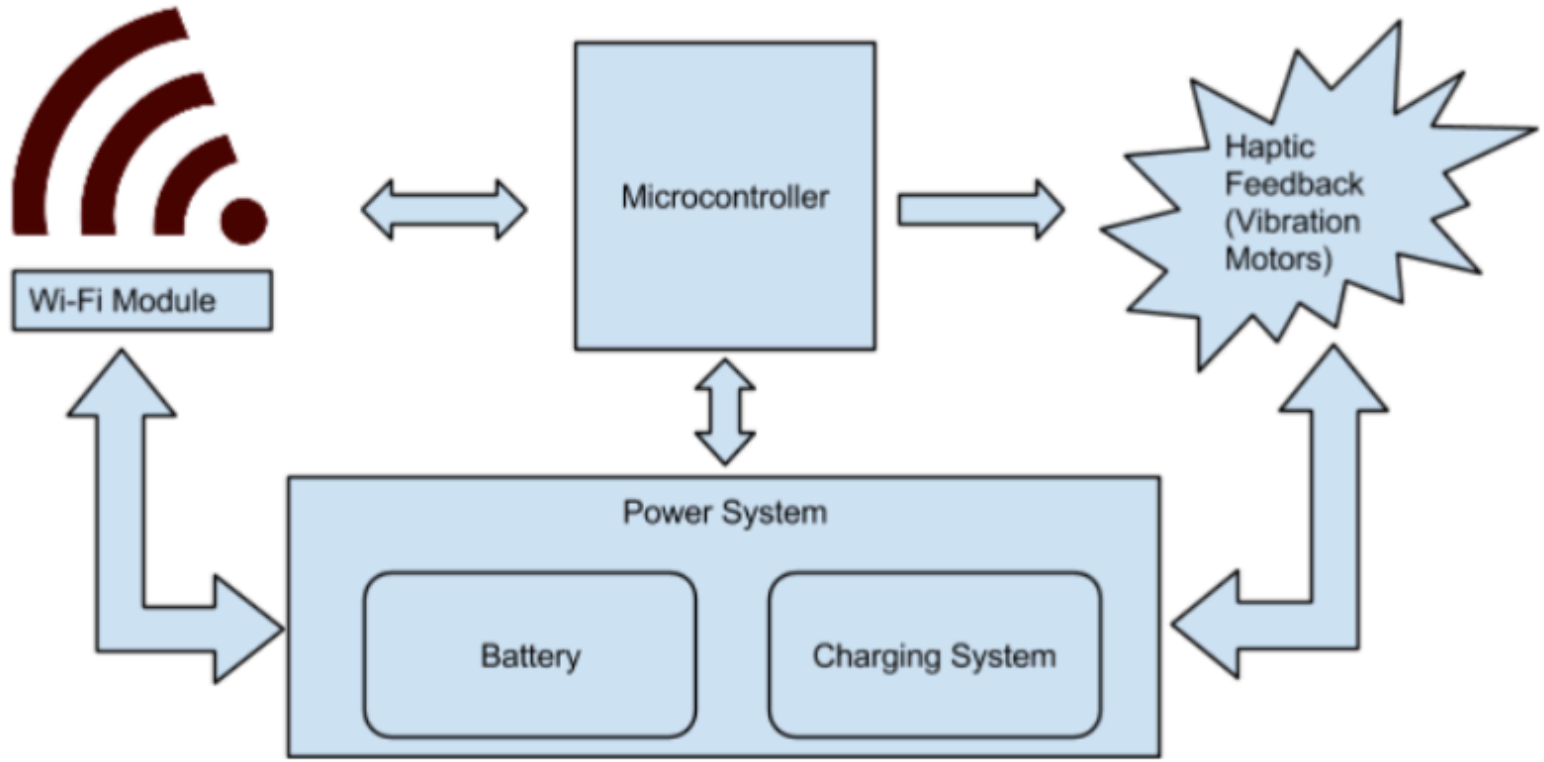


Figure 2: Overview of wristband featuring our 4 subsystems

Block	Requirement	Verification	Points
Module 1: Video Recording	Records video at a minimum of 12fps	One minute of video footage will be recorded on the glasses camera and extracted The video will then be imported into a MATLAB program for analysis Using the VideoReader() function, the program checks if the number of frames are greater than 720 which means the video is shot with at least 12 fps	5 pts

Design: Intellectual structure

Problem							
Requirement 1			Requirement 2		Requirement 3		
Req. 1.1	Req. 1.2	Req. 1.3	Req. 2.1	Req. 2.2	Req. 3.1	Req. 3.2	Req. 3.3
Logical Integration							

Design: Example

Social-emotional agnosia

(inability to read facial expressions)

Capture 90% of facial expressions from people user is talking with.			Classify 5 basic emotions with 95% accuracy using facial expressions.		Provide haptic feedback to user.		
Record video at 12 frames per second or faster.	Activate camera when faces detected at distances of 0.3m to at least 2m with greater than 91% accuracy.	Transmit each image of each facial expression over Wi-Fi with 99% accuracy.	Classify five basic facial expressions and neutral with 95% accuracy.	Classify facial expressions in less than one second.	Provide between 0.25-0.5N force to user.	Provide feedback in less than 50ms of receiving label.	Require less than 50mA of current.

Logical Integration

Design: Example

Social-emotional agnosia **(inability to read facial expressions)**		
Capture 90% of facial expressions from people user is talking with.		
Record video at 12 frames per second or faster.	Activate camera when faces detected at distances of 0.3m to at least 2m with greater than 91% accuracy.	Transmit each image of each facial expression over Wi-Fi with 99% accuracy.
Logical Integration		

Design: Example

<table>
<tr><th colspan="8">Solution</th></tr>
<tr><td colspan="3">Verification 1</td><td colspan="2">Verification 2</td><td colspan="3">Verification 3</td></tr>
<tr><td>Ver. 1.1</td><td>Ver. 1.2</td><td>Ver. 1.3</td><td>Ver. 2.1</td><td>Ver. 2.2</td><td>Ver. 3.1</td><td>Ver. 3.2</td><td>Ver. 3.3</td></tr>
<tr><td colspan="8">Debugging</td></tr>
</table>

Design: Testing and verification

Build a facial expression prosthesis

Build database of 20 conversations (vary gender, length, distance from camera). Manually label each facial expression. Play videos back to camera system. Compare number of expressions recorded by system to ground truth. Report accuracy.

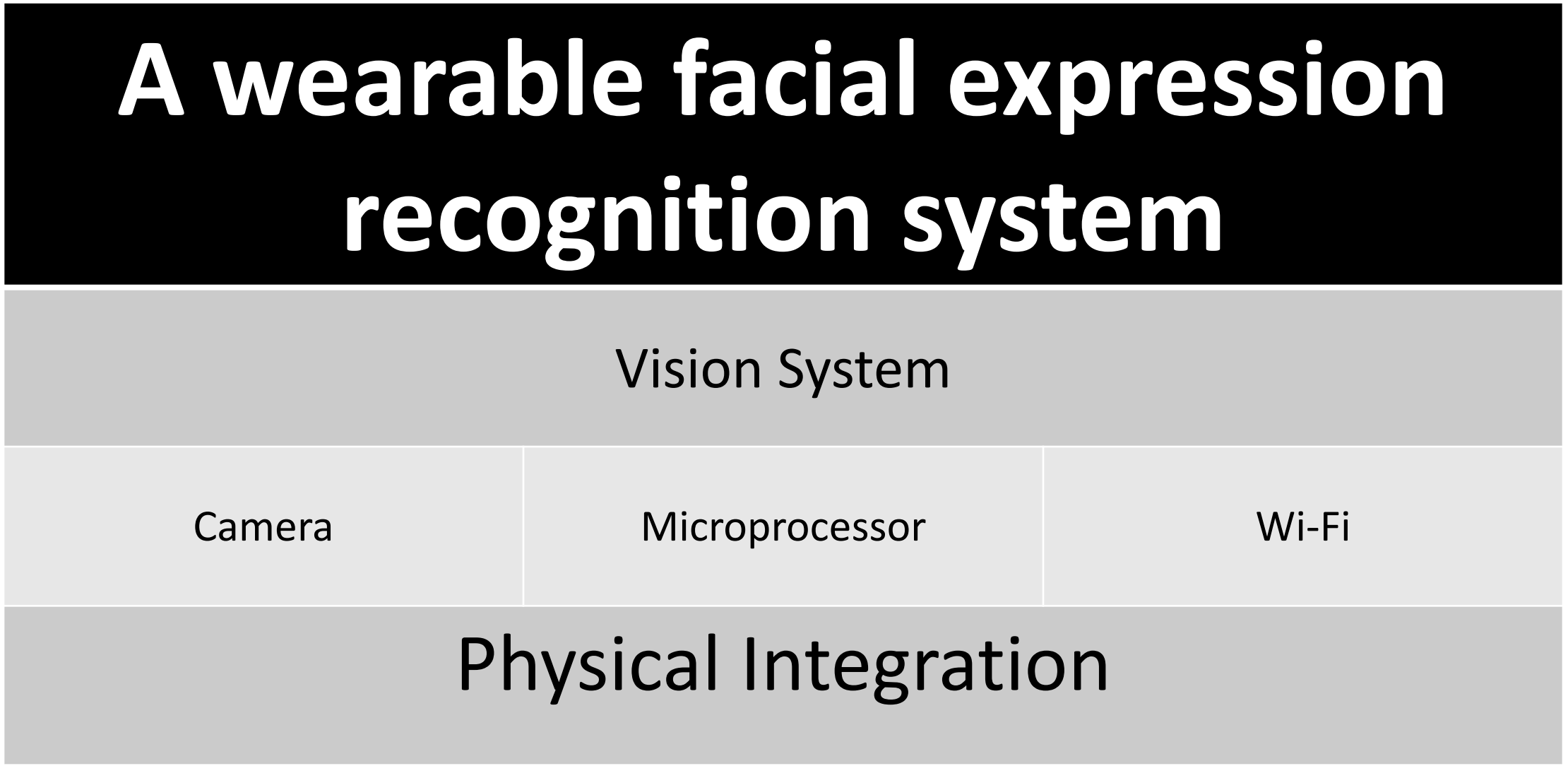
Record video of stopwatch with camera for 5 seconds. Does 5 second clip includes 60 frames? Repeat 100 times. Report count of frames for each trial.	Create training data set of 1000 images. Half images with faces, half without. Vary gender and distance (0.1-3m) from camera. Record number of times camera is activated by face images. Report accuracy.	Transmit 1000 images over the Wi-Fi connection. Save transmitted images. Compare saved images to original images. Report percent of pixels that match.

Debugging

Design: Correspondence to physical realization

<table>
<tr><th colspan="7">Product</th></tr>
<tr><td colspan="3">Module 1</td><td colspan="2">Module 2</td><td colspan="2">Module 3</td></tr>
<tr><td>Component 1.1</td><td>Component 1.1</td><td>Component 1.3</td><td>Component 2.1</td><td>Component 2.2</td><td>Component 3.1</td><td>Component 3.2</td><td>Component 3.3</td></tr>
<tr><td colspan="7">Physical Integration</td></tr>
</table>

Design: Example



The 445 report game…

Let’s pick a project at random and think about what they did well and what they could have done better in terms of their:

1. Block diagram
2. Block description
3. Requirements
4. Verifications

As a reminder, this is NOT an exercise to personally judge anyone’s work, but is analogous to performing a design review of a project.

Let’s (re)invent something!

- Pick an every day problem or task
- Deconstruct the problem
- Write requirements for a solution
- How could you verify the solution?