

About Azman

"One should not pursue goals that are easily achieved. One must develop an instinct for what one can just barely achieve through one's greatest efforts." - **Albert Einstein**

 Curriculum Vitae: [Basic Resume \(Mini\)](#), [Google Scholar Profile](#)

Code repository: [Main](#), [OLD](#), [MY1](#)

Feature: [my1sim85 \(releases\)](#), [my1digitaljs \(try online\)](#), [Mini-blog](#)

Academic Courses: [NMK322 - Microcontroller](#), [NMK201 - Microprocessor](#), [NMK206 - Computer Architecture](#), [PGT444 - B.Eng.Tech. Project \(FYP @FTKE\)](#)

Archived Academic Courses: [PGT302 - Embedded Software Technology](#), [PGT104 - Digital Electronics](#), [PGT206 - Computer Architecture](#), [PGT200 - Operating Systems](#), [EKT322 - Embedded System Design](#), [EKT222 - Microprocessor Systems](#), [EKT422 - Computer Architecture](#), [EMT251 - Introduction to IC Design](#), [EKT444 - Final Year Project @SCCE](#)

Intro Note(s): [Code:C](#), [Code:C++](#), [Code:Python](#), [Code:Javascript](#), [Code:HDL](#), [System Design](#)

Useful Note(s): [Linux](#), [FreeBSD](#), [Raspberry Pi](#), [MCU51](#), [Git](#)

Academic Schedule

*"Education is what remains after one **has forgotten** what one has learned in school."* - **Albert Einstein**

Last Updated: 202510052228 (FET-7.5.1-2510051103)

	0800 - 0900	0900 - 1000	1000 - 1100	1100 - 1200	1200 - 1300	1300 - 1400	1400 - 1500	1500 - 1600	1600 - 1700	1700 - 1800	1800 - 1900
MON											
TUE							NMK322Lab				
WED											
THU	NMK201Lab (007)										
FRI								NMK322Lec			

Note: '*' denotes 30 minutes time shift, '%' denotes 30 minutes time delay (starting only for 1½h session)

Note: {NMK322Lab:FKTEN-MKP1_LAB} {NMK201Lab:FKTEN-MICROPROCESSOR_LAB} {NMK322Lec:DK10}

[Here is something I wrote](#) intended for new students

[Another something I wrote](#) on education/training (based on stuffs I found online)

Latest Mini-blog Post

*"Intelligence is not the ability to store information, but to know where to **find** it."* - **Albert Einstein**

[Read all posts](#)

Academic Advisees

*"Student is not a container you have to fill, but a torch you have to **light up**."* - **Albert Einstein**

Students who are very (un)fortunate to have me as their academic advisor.

Updated: 20250910

NAME	MATRIC #	PROGRAM
HELAZIA ANAK SAPERI	231371750	UR6523007
ISKANDAR TAJUDIN BIN ZAMANI TAJUDIN	231371751	UR6523007
LEONG HAO JIE	231371752	UR6523007
MOHAMAD HAZIQ AMRIMULLAH BIN AFFIZAN EADY	231371754	UR6523007
MUHAMMAD ANAS NAUFAL BIN RUSSLI	231371757	UR6523007
HAZIQ ZULFIKRI BIN ZAHARI	231373099	UR6523007
ERIC DOMINIC	231373097	UR6523007
AINUL NAJIHA BINTI MOHD RAZMIN	231373869	UR6523007
GAVVENDDRA A/L SIVA CHANDRA	231373874	UR6523007

Points to Ponder

*"Try not to become a man of success. Rather become a man of **value**."* - **Albert Einstein**

I have been thinking about creating an ideal (from my point of view) bachelor program for [Electronics Engineering Curriculum](#).

I found this post in my old Google Currents (was initially known as Google+):



20130620094302 Dear fellow educators... let us remember that we actually need to educate, not to teach the students how to answer questions. The students' grades are **THEIR** performance indicators. The tests and examinations are to evaluate the students and **NOT** the educators.

Here is another one:



20130911102906 There are many small scale microprocessor/microcontroller boards



these days that offer 'simple' development environment setup/usage. To top that up, a whole library of interface codes for many commonly used devices are also provided. If you're into developing small-scale hobby projects or prototyping simple systems, this is a GREAT thing. But, if you're from computer/electronics engineering background and working in an academic institution, you're expected to know how to DEVELOP that board - not just use it. Think of what your students can learn if you strive to develop your own development board!

Bloom's Learning Taxonomy

*"Any fool can know. The point is to **understand**."* - Albert Einstein

My personal view on this.

Note My main reference regarding this is the [Wikipedia](#) page, and this other [site](#).

What It Is

The original learning taxonomy by Benjamin Bloom (1956) have identified three learning domains: **Cognitive** (mental skills @ knowledge), **Affective** (growth in feelings or emotional areas @ attitude) and **Psychomotor** (manual or physical skills @ skills). Bloom (and his committee) elaborated only on the first domain, with the second one defined by another group that also include Bloom (Krathwohl, Bloom, Masia, 1973). The definition of the third domain, however, was not directly linked to Bloom. Instead it has been discussed in 3 popular versions: Simpson (1972), Dave's (1975) and Harrow's (1972).

Note: Lorin Anderson, a former student of Bloom, revisited the cognitive domain in the learning taxonomy in the mid-nineties and made some changes, with perhaps the two most prominent ones being, 1) changing the names in the six categories from noun to verb forms, and 2) slightly rearranging them (Pohl, 2000). The name changes has been included below after the original names and in bold italic. The 'create' and 'evaluate' categories have been swapped and this is thought to be a more accurate hierarchy.

In the latest development, I can see the rationale of Anderson's arrangements where synthesis (create) is placed highest in the hierarchy. However, I think Bloom's decision to place it where it were (below evaluation) is simply based on a different point of view. The way he sees it, you cannot possibly make an evaluation if you cannot produce (synthesis) something of your own - i.e. going through the experience. So, in that sense, the original Bloom's Taxonomy was, I think, spot-on. Then again, nowadays a lot of people claim themselves to be expert (making critical reviews of other people's work) when they themselves have never actually had the experience! Go watch [Ratatouille](#) and you'll understand what I mean.

So, bottom line? Personally, I think synthesis should be considered at a higher level so as to remind others that the people who do evaluation needs to actually gain that level (synthesis) before they can evaluate synthesized work! So, in a way, that's also saying evaluation is higher than synthesis. Therefore, I would place them at the same level and only those who are excellent in respective fields can acquire this level. We should not expect EVERYONE to attain them.

Note: Just a thought - looking at Einstein's quote, notice that the point is to **understand**! So, in my

opinion, Bloom's level 2 is good enough for general education objective. The higher levels are relatively 'expert' levels that should not be imposed on everybody. Of course, this assumes the evaluation is done correctly (because the way things are done, even the so-called cognitive-level-6 evaluation is based on things that can be memorized).

Cognitive Domain

No matter what others say, I think this is the only domain that directly affects the learning process. This is where it all begins, and this is where it grows. So, I personally think this domain is still very much important and needs to be emphasized especially in technical education where most people thinks the psychomotor domain should be given 'equal' attention, along with affective domain. This really depends on how they define these two domains. I think Bloom was right not to define the two 'support' (at least from my point of view) domains. Although I don't deny the importance of affective domain in ensuring proper transfer of knowledge, I really think this domain boils down to ONE thing: how interested (or, maybe how motivated - but this can be discussed elsewhere) are the learners in obtaining the knowledge. And how high (or from another point of view, how deep) are they interested in going? (i.e. just want to know, capable of explaining to others, capable of improvising, etc.) On the other hand, psychomotor is really just how well our physical self can implement the knowledge. To me, all the 'advanced' levels in psychomotor are obtained through integration with the cognitive domain. (Maybe I should write a whole paragraph on this... later).

The six levels as described in wikipedia:

- **Knowledge (Remember)** ⇒ Exhibit memory of previously-learned materials by recalling facts, terms, basic concepts and answers
- **Comprehension (Understand)** ⇒ Demonstrative understanding of facts and ideas by organizing, comparing, translating, interpreting, giving descriptions, and stating main ideas
- **Application (Apply)** ⇒ Using new knowledge. Solve problems to new situations by applying acquired knowledge, facts, techniques and rules in a different way
- **Analysis (Analyze)** ⇒ Examine and break information into parts by identifying motives or causes. Make inferences and find evidence to support generalizations
- **Synthesis (Create)** ⇒ Compile information together in a different way by combining elements in a new pattern or proposing alternative solutions
- **Evaluation (Evaluate)** ⇒ Present and defend opinions by making judgments about information, validity of ideas or quality of work based on a set of criteria

Affective Domain

From my personal point of view - when it comes to learning, as I've already mentioned earlier, this domain actually displays interest level of the student. If a student is only in the class because he/she HAS to, they will only (at most) listen or **receive** what is being taught. They will not inquire more than what is being given.

Note: *maybe I can elaborate more later???*

The five levels in this domain as described in wikipedia:

- **Receiving** ⇒ The lowest level; the student passively pays attention. Without this level no learning can occur.
- **Responding** ⇒ The student actively participates in the learning process, not only attends to a stimulus; the student also reacts in some way.
- **Valuing** ⇒ The student attaches a value to an object, phenomenon, or piece of information.

- **Organizing** ⇒ The student can put together different values, information, and ideas and accommodate them within his/her own schema; comparing, relating and elaborating on what has been learned.
- **Characterizing** ⇒ The student holds a particular value or belief that now exerts influence on his/her behaviour so that it becomes a characteristic.

Psychomotor Domain

This domain is actually the 'physical' side of things. When this taxonomy was created, the examples given were the manipulation of tools and instruments (e.g. hand, hammer). So, translating that to the current scenario, that includes using computers - typing, using mouse/pointers, etc.

Rightfully so, Bloom and his colleagues never created subcategories for skills in the psychomotor domain. However, since then, other educators have created their own psychomotor taxonomies.

Outcome-Based Education

The buzz word of academic world in current years, the OBE method suggests that we should concentrate on the students' ability to perform (the outcome). This has sometimes been misinterpreted as saying that the 'previous' method does not have any objective and we actually have one when using this method. Well, the 'previous' method also have its objectives - I do not think there is anybody who wants to implement something without ANY predetermined objective. It should be looked from a different perspective.

The academic world has traditionally relied on timed written examinations as the main evaluation method. As we all know, some people do things slowly and some cannot work at all under pressure (time limits!) - but this does not mean that they cannot do the work. So, this is saying that we should acknowledge if a student can actually perform the same tasks amid being slower or requiring a more peaceful environment. In other words, we should be evaluating the students ability to actually perform a task (an outcome), rather than their ability to memorize.

Bottom line, I would say that in OBE, we should be looking for a **suitable evaluation method** for each outcome (usually an ability) rather than discussing things like course work marks contributions, or what cognitive level should a question be assigned to, or even how much time should be spent on tutorials. Then, the evaluation should REALLY reflect the students ability (the outcome) and there should not be a case where a student can get an F in circuit theory when he/she can actually perform impedance matching, or a student who get an A for C programming, but has never written a complete program in C.

SMART Objectives

Note: I wrote this a while back, just thought I should share it here. The current 'official' page for this is [here](#).

Having to go through OBE exercises, I came across this term - which really interests me. It's an acronym that can be used as a guide when trying to set objectives or goals. They can be project objectives, company objectives, or (in my case) course objectives.

So, **SMART** objectives are:

- **Specific** - Clear and well-defined
- **Measurable** - So that we know when it's been completed and how well it's been done
- **Achievable** - To have closure and accountability
- **Relevant** - Must be useful (e.g. part of global strategies) and within context (e.g. aligned with higher goals)
- **Timely** - Must have a projected time-line, consistent with being achievable and measurable (which stems from being specific)

Some also use **Realistic** for **R** - but I think it's redundant with achievable (from terminology point of view). Also, you can make it **SMARTER** if you can:

- **Evaluate**
- **Reevaluate**

I have also just come across [this](#). So, this guy is proposing **DUMB** objectives instead.

- **Dreamy**
- **Unrealistic**
- **Motivating**
- **Bold**

Some may put it differently (**Doable**, **Understandable**, **Manageable** & **Beneficial**), but I think that's just trying to be as **SMART**.

Just think about it - you may show that you seem to have done a lot with **SMART** objectives, but can you really be outstanding without **DUMB** (the dreamy one... ) objectives?

Other Interest

Things I'm checking out... 

- [RISC-V](#) - RISC open architecture
- [J-Core](#) - SH-compatible (SuperH instruction set) open core design
- [Tiny Tapeout](#) - For me, the new MOSIS :p
-  [Neuromorphic](#) - VLSI with analog circuits... an example is the [Neurogrid](#)
-  [Memristor](#) - Memory capable circuit element... better density, better suited for analog integration?
- [CoreBoot](#) - BIOS replacement
- [FlashROM](#) - Need this for coreboot?
- [XCore86](#) - i586-compatible processor that can be used in [systems](#) that run on AA batteries???
- [Wireless Power](#) - Wireless Charging?
- [Game Theory](#) - Something that I was VERY interested in, but just couldn't get the time to look further...
- [XL Game Engine](#) - Nice!

- [Open Simulator](#) - 3D Application Server
- [suckless.org](#) - what do you think?
- [Open Street Map](#) - Google Maps alternative? Data only?
- [Open Layers](#) - Mapping JS library (BSD licensed)
- [Matrix](#) - Open Network, Decentralized Communication
- <https://emscripten.org/index.html> - Compile C/C++ code into Javascript?
- [QuickJS](#) - Small MIT-licensed JS engine! ([GitHub](#))

Existing Boards (for comparison):

- [Gumstix](#) boards: [Overo Sand COM](#) costs US\$115
- [BeagleBoard](#): BeagleBone costs US\$89
- [Raspberry Pi](#): costs US\$25@US\$35

Reference:

- [pcb](#) - Tutorial for PCB layout software
- Wiring info: [Wiring](#)
 - basically, the minimum (AWG30-gauge) is said to achieve 1A easily although the above site states 860mA/142mA

Check Out These Boards?:

- [CUDA?](#) on ARM?
- [CEL MeshConnect](#)
- [Spartan3](#)

Interesting Links:

- [Image Completer?](#) - Not really related... just like to read about it.
- [Auto-Focus](#) - Something I want to explore... also [here](#)
- [IceStorm](#) - Reverse engineering project that allows fully open source Verilog-to-Bitstream flow for [iCE40 FPGAs](#)
- [IceStudio](#) - FPGA Block Diagram Design Tool (plus Programmer)... Cool! 😎
 - For [this \(manufacturer site\)](#)... that's double cool 😎😎
- [Webots](#) - Robot Simulator
- [Stage](#) - Multiple Robot Simulator (Part of the [Player Project](#)) ([Repo](#))
- [Gazebo](#) - 3D Multiple Robot Simulator (*[Used to be?]* Part of the [Player Project](#)) ([Repo](#))
- [Player](#) - Robot device Interface (Part of the [Player Project](#))
- [ARGoS](#) - Large-scale Robot Simulator?

Information on various open source license(s) at [OSI website](#).

I'm inclined towards the [MIT](#) license and the [Simplified \(2-clause\) BSD](#) license. 📖 [Read more @ wikipedia](#)

Others:

- [Games Development](#) - From Scratch?
- [SFML](#) - Compared with [SDL](#)?
- [Using hosts file](#) - Interesting!
- [InfraRed DB](#) - May want this as reference?

Note on Development Platform

I have decided to use Linux exclusively for my project development work. Which Linux distribution should I use? Let's ask Einstein...



Note #1: I got this image from a link in an [LQ](#) forum thread. Please inform me if I should NOT use this image (from legal point of view).

Note #2: I would also like to clarify that you do NOT have to be Einstein to run Slackware. But, you DO need to know how to read (instructions) and write (specifically, type some stuffs on a computer).

All development work will be done on Linux-based systems - [Slackware](#) is recommended and is what I use ([Devuan](#) is also a good substitute and what I setup for my family), but I'll also try to accommodate people working on other platforms.

Links to local pages

Training Courses: [Introduction to Verilog and ModelSim](#), [Using Raspberry Pi as Data Server for Embedded Systems](#)

Useful Notes: [8051 Stuff](#), [Raspberry Pi Stuff](#), [FreeBSD](#), [Linux \(Slackware| Devuan\)](#), [Android App Development](#), [Git \(SCM\) Stuff](#), [Vim Stuff](#), [Sed Stuff](#), [Database Stuff](#), [Screen Stuff](#), [Qemu Stuff](#), [LibreOffice Stuff](#), [MathPublish\(Dokuwiki\) Stuff](#), [Cross Compiler Stuff](#), [Samba AD Domain Controller](#), [Cisco Stuff](#), [Android Stuff](#), [VirtualBox Stuff](#), [Python CV Stuff](#), [Scripting Stuff](#), [Htaccess Stuff](#), [Random Stuff](#), [Sync Google Drive on Linux](#) FPGA Notes: [Xilinx](#), [Xilinx Spartan-3 VSK Barebone](#), [Altera Dev](#).
Notes: [Board\(s\)](#), [Display\(s\)](#)

Basic Intro Stuff: [Basic Programming in C](#), [Introduction to Programming in Python](#), [Basic Javascript Programming](#), [Introduction to HDL](#), [Basic System Design](#)

Archive: [Nokia 3310 LCD](#), [Pioneer-3 Robot SDK Basics](#), [Window\\$ Stuff](#), [Linux on Android](#)

Mini Info-pedia

This is a personal information center meant to help explain some of the terms that I use - from my point of view.

Development Board

A **development board** is meant to help system developers to learn the inner workings, interfaces and features of a certain processor. Thus, it usually contains minimal components that are required to start up a system based on that certain processor. However, many development boards are currently equipped with features comparable to that of a desktop computer's motherboard. This allows them to be used in virtually any test scenarios due to the fact that they provide basic input and output interfaces that are easily available off-the-shelf.

Many will be more familiar with **microprocessor development board** instead, but since we have other processing units such as Digital Signal Processors (DSP) and Field Programmable Gate Arrays (FPGA) that are also available on boards with similar functions, it would be more appropriate to generalize the term.

With the ever-increasing features of development boards, many scientific research have started to use them as a system platform.

All that been said, development boards are typically intended for prototyping and not usually used in an end product.

 [Read more @ wikipedia](#)

Embedded System

An **embedded system** is a combination of hardware and software (sometimes the term *firmware* is used as well) components that is usually meant to be part of a larger system (thus the word *embedded*). However, it is technically a standalone system (one that works without any external support) and is primarily designed for a specific purpose (i.e. with predefined tasks and dedicated functions). With this in mind, an embedded system requires minimal user input (minimal user interface, not re-programmable by end-users) and provides only the expected output depending on its application.

Since an embedded system only needs to do a specific task, it can most of the time respond to real-time events (system input). The occurrence of these 'real-time' events can even be predicted due to the fact that a predefined task usually have a known set of inputs. Based on this scenario, an embedded system does not (most of the time) require a state-of-the-art central processing unit (CPU) to operate. Consequently, the preferred CPU for an embedded system is a microcontroller, which is basically a microprocessor that emphasizes self-sufficiency (e.g. on-chip memory, minimal glue logic) and therefore offers a cost effective solution.

Current implementations of embedded systems are made programmable so that the main system developers can easily make appropriate upgrades or modifications. This enables modern embedded systems to have more features (upgradable at any time) at a lower cost (none or minimal hardware changes).

 [Read more @ wikipedia](#)

Computer Vision

Research work related to vision comprises many different field of studies - each of which have different point of view on how it can be achieved and what feature should be emphasized. Therefore there a few variations to the name of this particular field of study - but they all actually try to achieve the same thing. It is mainly referred to as computer vision in the field of computer science, while the term machine vision (or robot vision) is more popular in the engineering discipline. Another familiar name would be computational vision, where interdisciplinary researchers get together to understand biological vision and create a system that can 'see'.

Computer vision is closely related to image processing and therefore a sufficient understanding of image processing techniques is required. Naturally, it is also the inverse of computer graphics and in fact, a substantial amount of work on vision systems have used graphics accelerator (graphics card) as the processing platform. Computer vision also requires some form of artificial intelligence, which separates it from basic image processing.

As defined by Marr (1982), vision is actually the process of knowing what and where an object is in its environment (something like that.. I'll refine this later). So, the basic tasks pursued by vision is image understanding (what it is), image analysis (where it is) and scene analysis (environment). The full flow can be further divided into 2 levels:

- early vision
 - low-level
 - image processing?
 - feature analysis/extraction, image segmentation
- scene analysis
 - high-level
 - knowledge-based processing
 - shape analysis, object recognition & localization

Main reference:

```
@book{993688,  
  author = {Edwin D. Reilly},  
  title = {Concise Encyclopedia of Computer Science},  
  year = {2004},  
  isbn = {0470090952},  
  publisher = {John Wiley & Sons},  
}
```

 [Read more @ wikipedia](#)

Temporary Note(s)

Things I cannot quite place anywhere else...

Something...

[Something](#) to remember...

Note to self

My `index.html` so that my unisite url always get redirected to my wiki page

[index.html](#)

```
<html><head>
<meta HTTP-EQUIV=REFRESH CONTENT="0;URL=dokuwiki/doku.php">
</head></html>
```

Note: The [mathpublish](#) plugin does not work on my current server - so, need to check requirements...
{tokenizer library|gd library(png (libpng,zlib)|freetype font library}

Dokuwiki Stuffs: [purge=true \(force re-caching\)](#) , [do=check](#) , `do=export_html` , `do=export_pdf` (plugin)

Setting up dokuwiki...

```
git clone https://github.com/splitbrain/dokuwiki.git
git checkout --track origin/stable
```

Dokuwiki plugins I use:

[dokuwiki_plugins.txt](#)

```
- wrap> https://www.dokuwiki.org/plugin:wrap
- include> https://www.dokuwiki.org/plugin:include
  = tag> https://www.dokuwiki.org/plugin:tag
- folded> https://www.dokuwiki.org/plugin:folded
- blog> https://www.dokuwiki.org/plugin:blog
  = pagelist> https://www.dokuwiki.org/plugin:pagelist
- dw2pdf> https://www.dokuwiki.org/plugin:dw2pdf
- iframe> https://www.dokuwiki.org/plugin:iframe
- math2|mathpublish> https://www.dokuwiki.org/plugin:mathpublish
- pagebreak> https://www.dokuwiki.org/plugin:pagebreak
- todo> https://www.dokuwiki.org/plugin:todo
- vshare> https://www.dokuwiki.org/plugin:vshare
```

From:

<http://azman.unimap.edu.my/dokuwiki/> - **Azman @UniMAP**

Permanent link:

<http://azman.unimap.edu.my/dokuwiki/doku.php?id=start&rev=1760271890>

Last update: **2025/10/12 20:24**

