Abdulrahman **Yasser Abdulrahman**

abdulrahman.yasser74@gmail.com | +201123400632 | Egypt, Cairo | Military Service Completed
https://github.com/Abdulrahman-Yasser/ | linkedin.com/in/abdulrahman–yasser/
Medium | Instructables | Certificates | Youtube

Education

Obour High Institute

BSC in Computer Engineering and Control Systems — *Jul 2022*

Graduation project: Self-driving car implementation - grade: Excellent - No sponsorship.
Used many image processing algorithms to enhance the taken images. Created, trained, and evaluated many deep learning and computer vision models. I've used TM4C123 and Raspberri-pi4 in the project.

Experience & Courses

Data structure and Algorithms in Cpp - Dr. Mostafa Saad - Udemy — *Github*

Data Structure, Algorithms 1, and Algorithms 2 — *Mar 2024 – present*

- More than 55 hours learning how to solve many data structures and algorithms problems
- Learned and solved problems on Recursion, complexity analysis, sorting, searching, graph representation, graph depth-first search, graph breadth-first search, and topological sorting.
- Learned and solved problems on Dynamic Programming, and many graph algorithms including Floyd-Warshall, Bellman-Ford, Dijkstra, and many others.
- Learned and solved problems on Vector, Linked List, stack, queues, AVL tree, and Hash tables.

Embedded Linux Course — *Github*

Under the supervision of Eng.Moatasem El-sayed — *Mar 2024 – present*

- Building kernel with my configurations
- Building a toolchain using Crosstool-NG .
- Controlling booting an Image using U-boot, and serial booting, and TFTF.
- Communication protocols through Raspberry-pi and Linux like xmodem, ymodem, and other protocols.

Advanced Embedded Systems — *Github*

Future Work is Digital (FWD) and Udacity — *Aug 2022 – Oct 2022*

- Learned about ARM and Build Processes, ARM Cortex-M Architecture, ARM Exceptions and Interrupts, NVIC, System Control and Clock, GPIO, WDT, ADC, and GPT. It concluded with a graduation project where I developed GPIO, NVIC, GPT, and SysTick drivers to control an LED On-Off time in run-time by varying a PWM signal.
- RTOS Masterclass, Inter-Process Communication, Design of real-time Systems, Schedulers, Run-time analysis, RTOS issues, and fixes and porting.Concluded (EDF) scheduler and verified it by designing and simulating a real-time system of 6 inter-communicating tasks on SimSo and Keil uVision.
- Embedded Software Static and Dynamic Design, Modular Programming, Development Life Cycle, Layered Architecture, Module Design, UML, and Design Patterns. Concluded by a full static and dynamic design of an Automotive Door Control System based on CAN.
- Safe Coding MISRA C, MISRA, Foundations of Agile and Software Testing .

IOT & AI — *Github*

NTI — *Jul 2022 – Aug 2022*

- Created, trained, and modified many Computer vision and Deep learning models.
- Made a mobile application using node-red connected with the MQTT network for transmission and reception.
- My graduation project was a car's plate number reader, which receives the target number from ESP8266 over MQTT, searches in its dataset for that particular car based on the plate number, and responds with a message over MQTT.

Embedded Automotive and AUTOSAR Device Drivers Course — *Github*

SignEd course center under the supervision of Eng.Mohammed Tarek — *Oct 2021 – Jan 2022*

- AUTOSAR Layered Architecture, Device Drivers, C Misra Rules. Automotive buses Lin and Can.
- Implement Dio and Port AUTOSAR Driver for TM4C Micro-controllers as my final project to apply the full layered architecture model.
- ARM CortexM3/M4 MPU, GPIO, PLL, and SysTick Timer Drivers. NVIC System: (Edge Triggered Interrupts), System Exceptions: (PendSV, SVC, and SysTick Exceptions), and Fault Exceptions: (HardFault, UsageFault, BusFault, and MemoryManagement Fault).

Data Structure and Algorithms by BERKELEY CS61B — *Github*

3 Months of studying data structure and algorithms in Java — *Jul 2022 – Oct 2022*

- Project 0 - 2048: building the core logic of this game.
- Project 1 - Guitar Hero: build implementations of a "Double Ended Queue" using both lists and arrays in a package that other classes can use.
- Project 2 - Gitlet: implementing a version-control system that mimics some of the basic features of the popular system Git.

Small Hero

Teacher — *Jul 2020 – Aug 2020*

- I've taught over 50 kids in the academy how electricity works and how to use Arduino using mBlock ide.

Technical Skills

Programming Languages:	C, C++, Python
Communication protocols:	SPI, I2C, LIN, CAN, UART, Wifi
Embedded SoC:	TIVA-C, Raspberry-pi, AVR, ESP8266
Embedded tools:	Atmel studio, IAR, PulseView, CCS, PuTTY, Logic analyzers, ESP-IDF
Python Libraries:	pySerial, OpenCV, TensorFlow, Pandas, virtual environment, Selenium
Good Knowledge of RTOS:	FreeRTOS, Time-Triggered Patterns
Good knowledge of Linux:	Experience in development on UNIX/Linux platforms.
Good knowledge of Embedded Linux	Yocto, Device Driver, Bootloaders (U-Boot), Building Kernel Building Toolchains (Crosstool-NG), Busybox, Bash Script.
Basic knowledge of Computer Vision:	CNN, TensorFlow, Google Colab
other tools:	Git, Cmake, Doxygen, QEMU
Good knowledge of Design Patterns in C	
Good knowledge of Layered Software Architectures	
Basic knowledge of AUTOSAR	

Projects

Self-driving car implementation (Graduation project - Excellent)

Google Colab, TensorFlow, OpenCV, YOLO-V3, Behavioral cloning, Tiva-C, Raspberry-Pi

- The hardware was a TM4C controlling the motors using a PWM module and communicating with the Raspberry Pi via UART. Applied image processing techniques to calculate the road's curvature and clarify the captured image for future processing. Implemented and trained Nvidia CNN behavior cloning architecture. Created, trained, and evaluated a CNN model for traffic sign classification. Implemented and trained YOLOv3 for object detection on the road. Merged two datasets to increase the object numbers that can be detected by the model (YOLO).

I've run most of the trained models on raspberry-pi4 to check the results.

Github

Locate that Car

Google Colab, TensorFlow, ESP8266, HiveMQ

- AI & IOT Course's project was a v2x. Modified and trained a yoloV3 model to detect car plate numbers (Vehicle). The vehicle connected with a Nodemcu (X) to receive the desired plate number and return a response if it has been found.

Github

Layered Software Architecture Drivers (Bare-metal)

Tiva-C, PORT, UART, SPI, CAN, I2C, NVIC, LCD, Seven Segment, RTC, ...

- More than 12 Module Drivers on the Tiva-C. My software architecture uses four layers: Common, HAL, MCAL, and APP. All the drivers are divided into the Static part, which shouldn't be changed, and the Dynamic part, which has the Configuration changed by the users

Github

Applications based on UML diagrams and Design-Patterns

Tiva-C, Observer, Client-Server, Singelton patterns, UML diagrams, LCD, Sensors

- Wrote, modified, and ran many codes from the book "Design Patterns for Embedded Systems in C by Bruce Powel Douglass".
- Weather station applications using Client-Server, and Observer patterns .
- Polling, Interrupting, and debouncing push-button applications based on UML diagrams.
- Sensor class using Observer Pattern "Publish-Subscribe Pattern" that provides notification to a set of interested clients that relevant data have changed, and clients can be changed in run-time.
- UML diagram for Static priority RTOS Controlling motor application.
- Motor position reading using critical region and Guarded call patterns to limit the code access.

Github

Time-Triggered schedulers.

Pre-emptive, Co-operative, Hybrid schedulers, Multi-State, Multi-Stage applications LCD, Keypad, UART, I2C

- Wrote, modified, and ran many codes from the book "Patterns for Time-triggered Embedded Systems by Michael J. Pont" on Tiva-c. Created Co-operative and Hybrid schedulers.
- Controlling a flashing LED based on multi-state patterns .
- I've made multi-state and multi-stage applications using I2C, LCD, Keypad, and Tiva-C.

Github

Smart home using Espressif (ESP-IDF)

Nodemcu, ESP-IDF, RTOS, Wifi, BLE, NVS, Git, Visual Studio, Cmake, Cpp17

- Here I've adopted the best practices to make my code clean and efficient.
- Wrote WiFi, SNTP, NVS, and Bluetooth drivers using ESP-IDF APIs.

Github

Reusable Firmware Drivers

Git, Visual Studio, Cmake, Doxygen, Tiva-C, SPI, DIO

- These projects are based on the book "Reusable Firmware Development by Jacob Beningo". All the drivers are portable, reusable, readable, adaptable, and well-documented Bare-metal drivers.
- Based on the information and instruction in the book, I've created GPIO and SPI drivers

Github

Automation scripts

Python, Spotipy, Pytube, Selenium

- Automated YouTube and Spotify libraries to download any new song I add.
- I've written many web scraping codes to collect data from different websites.

Github