

1. Introduction to Team-Based Learning (TBL)

Team-Based Learning (TBL) fosters collaboration, problem-solving, and critical thinking. Unlike traditional labs with step-by-step guidance, TBL promotes teamwork, self-reliance, and competition. This session features a competition where students apply knowledge to solve tasks independently, with minimal TA intervention. Awards will recognize the best teams, reinforcing independent learning.

2. Team Formation

Each team consists of:

- **1 Team Leader:** Oversees task delegation and coordination.
- **1-2 Programmers:** Responsible for coding and firmware development.
- **1 Researcher:** Gets information from AI chatbot.
- **2-3 Reviewers/Testers:** Debug, validate, and document implementation.

All students are required to bring their own laptop computers to the TBL class.

3. Technical Background

A USB memory stick comprises a **USB controller, NAND Flash controller, and an ARM processor** connected via **AMBA AHB/APB buses** for communication.

3.1 Core Components

- **ARM Processor:** Executes firmware; contains ROM (boot code) and RAM (runtime execution).
- **USB Controller:** Manages USB communication; includes a DMA controller for efficient data transfers.
- **NAND Flash Controller:** Interfaces with NAND Flash memory; includes its own DMA controller for high-speed transfers.
- **Memory Buffers:** Temporary storage areas for USB, system operations, and NAND Flash data.

3.2 Data Flow Summary

1. The **USB controller** stores incoming data in the **USB buffer**.
2. The **DMA controller** moves data from the **USB buffer** to the **NAND Flash buffer** via **AHB bus**.
3. The **NAND Flash controller** writes data to **NAND Flash memory**.
4. For read operations, the process is reversed.
5. The **ARM processor** oversees all the firmware operations, processing read and write SCSI commands, converting logical to physical addresses, supporting data encryption, error correction, and power management, as well as handling bad blocks and other exceptional cases.

4. Tasks

Mandatory Tasks

- **Task 1:** Set up the SDK environment, compile the standard firmware, program the memory stick, and test functions.
- **Task 2:** Modify the C code to toggle the LED every second. **Before programming the memory stick, have the TAs review your source code.**
- **Task 3 (Competition):** Develop firmware to create two partitions. The first partition should always be accessible, while the second partition's visibility toggles each time the memory stick boots—visible on the first boot, hidden on the second, visible on the third, and so on.

Optional Task (Competition)

- **Task 4:** In the previous task, the second partition may be visible after multiple boots. This is not secure. Develop your **own proprietary secure memory stick** so that the second partition is always hidden unless the user knows the secret. You may take advantage of the LED light. Please make a **PowerPoint presentation** to describe how your secure memory stick works.

Important Hints:

- You need to modify `custom_setup()` and `custom_loop()` in `custom.c`.
 - `custom_setup()` runs **once** at boot time.
 - `custom_loop()` **continuously** runs in the foreground.
 - Both functions must execute quickly (under 100 microseconds); otherwise, the memory stick firmware may crash.
- Relevant APIs for LED and timer control:
 - `int gpio_configure(int pin, int direction, int initial_value)`
 - `void gpio_set(int pin, int value)`
 - `int gpio_get(int pin)`
 - `uint32 timer_get_raw(void)`
- All variables are reset upon boot. Hence, all non-volatile state variables must be saved in the user flash space. User flash space APIs:
 - `int user_data_set_bank(int bank_index)`
 - `int user_data_get_byte(int offset)`
 - `int user_data_set_byte(int offset, U8 value)`
 - `int user_data_flush(int force_flush)`
- At boot time, you have the option to mount or unmount partition #1. API for partition management:
 - `void system_mount_partition(int partition, int mount, int persist)`

5. Session Structure

0:00 - 0:15 | Introduction & Setup

- Overview of session goals and evaluation criteria.
- Distribution of resources (firmware template, documentation, software tools).

0:15 - 2:30 | Cycling Drive Visibility Competition (Mandatory)

- Implement partition cycling logic.
- Debug and test the firmware.
- Early finishers can optimize code or assist peers.
- **First team to complete receives an award and extra credit.**

0:15 - 2:30 | Secure Drive Development (Optional)

- Implement a hidden partition feature with LED unlock indication.
- Design security measures and document findings.
- **Best design and cleanest code earn an award and extra credit.**

2:30 - 3:00 | Evaluation & Debrief

- **Live Demonstration & Presentation:** Selected teams present solutions.
- **Judging Panel:** Evaluates functionality, creativity, and efficiency.
- **Award Ceremony:** Recognition of top teams.
- **Q&A and Closing Remarks.**

6. Evaluation Criteria

Criteria	Toggling Drive Visibility (Mandatory)	Secure Drive (Optional)
Functionality	50%	40%
Code Efficiency	20%	30%
Creativity & Innovation	10%	20%
Documentation	10%	10%
Teamwork & Debugging	10%	10%

7. Resources Provided

- Standard firmware template.
- ARM VLS5100 documentation.
- ARM VLS5100 SDK User Guide:
 - Build environment setup.
 - GPIO, SPI, and logical media operations.
 - Security features (password authentication, encrypted partitions).

8. Expected Outcomes

1. Hands-on experience in firmware development.
2. Enhanced problem-solving and teamwork skills.
3. Understanding of partition management and security in USB drives.