Security System: Vision & Bluetooth–Based Human Detection
ECE 5436/6336 Final Project Report

Team 12: Luke Dvorak, Jonathan Gaucin

***Abstract*—We present *MotionIoT*, an end-to-end security system that couples a CNN-based vision pipeline on a host PC with a Tiva-C microcontroller running a home-grown RTOS. Person-detection events are relayed over an HC-05 Bluetooth link. The embedded node provides local alarm feedback via LCD graphics and buzzer chirps. The project demonstrates wireless MCU/PC co-processing and rapid prototyping with open-source tools.**

I. Project Objectives

- Detect a human intruder within 2 m using YOLOv4-tiny on a laptop camera.
- Relay detection to a standalone MCU through Bluetooth (<100 ms).
- Fuse computer-vision and audio cues to reduce false alarms.
- Display state on EduBOOST-MKII LCD; chirp a buzzer for deterrence.
- Showcase a custom cooperative RTOS with four tasks and strict 1 kHz round-robin scheduling.

II. Functional Block Diagram

See Fig. 1, next page.

A. Functional Description

The RTOS consists of one **Main** routine plus four tasks that run in a round-robin list (Table I). Main initialises peripherals, draws axes on the LCD and starts SysTick.

Task 0 – Bluetooth / Alert: polls the UART FIFO for the ASCII value `'1'`. On the first hit it resets `DetCount`, flags `saw=1` and chirps the buzzer; it re-chirps every 1 s as long as detections continue.

Task 1 – Microphone: samples the MKII mic, computes an RMS window and sets `heard=1` if the level exceeds *SOUND_THRESHOLD* ten frames in a row.

Task 2 – UI: renders either the camera or microphone status line, plus "Alarm Disabled/Armed" overlays, and handles yellow/red detection bars.

Task 3 – Buttons: debounces Button 1 (mode toggle) and Button 2 (test-tone), and toggles the Arm/Disarm state when both are held for one second.

III. System Description

A. Hardware

Tiva-C TM4C123GXL
80 MHz Cortex-M4F running the RTOS.

EduBOOST-MKII
LCD, mic, push-buttons, RGB LED, buzzer.

HC-05
Classic Bluetooth module on UART1 (9600 baud).

Laptop (Win/Mac)
USB camera, Python 3.11, OpenCV 4.9.

B. Software Development Environment

- **Embedded:** Keil µVision 5, ARM CC, Segger J-Link.
- **Host PC:** Python, OpenCV-DNN, `pyserial`, YOLOv4-tiny weights.
- **Version control:** GitHub repo https://github.com/jgaucin03/MotionIoT.

IV. RTOS Design

The scheduler is a simple **co-operative round-robin** invoked by `SysTick_Handler` every 1 ms (1 kHz, equal quantum). Pre-emption occurs only at these tick boundaries; therefore we classify it as *non-preemptive round-robin*.

Task List

Pseudo-code Excerpt (Scheduler)

```
void SysTick_Handler(void){
    TCB_t *next = RunPt->next;    // round-
        robin list
    RunPt = next;
    (*RunPt->task)();             //
        cooperative switch
}
```

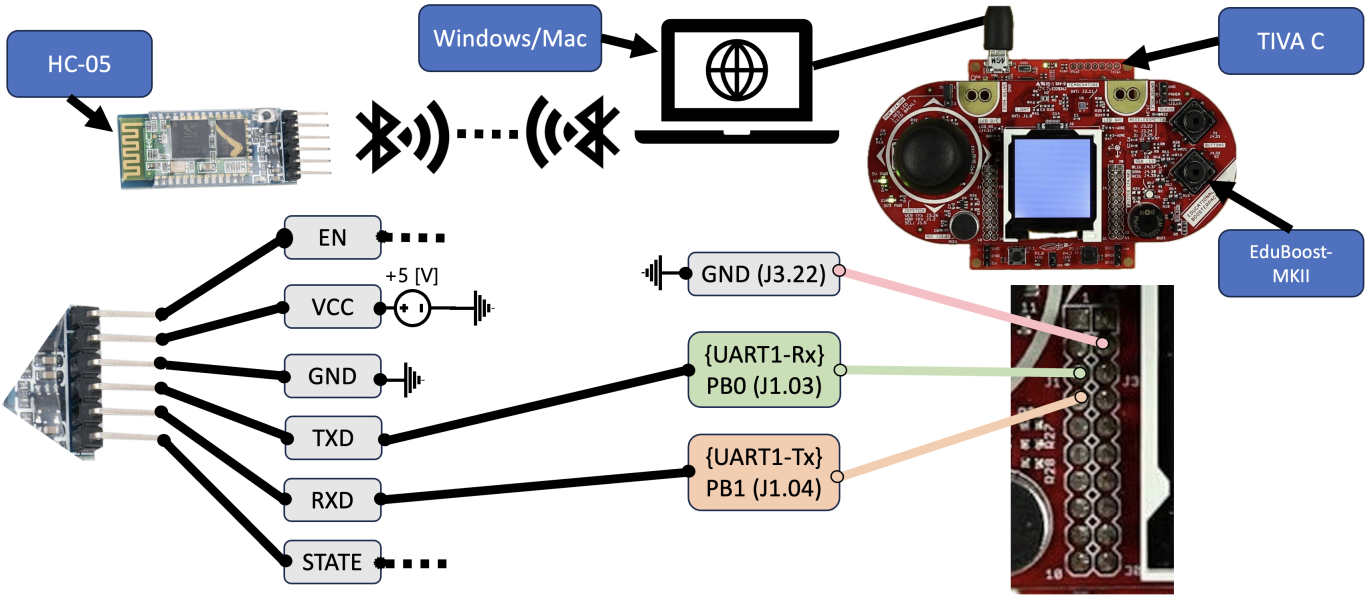

Fig. 1. High-level hardware/software partitioning.

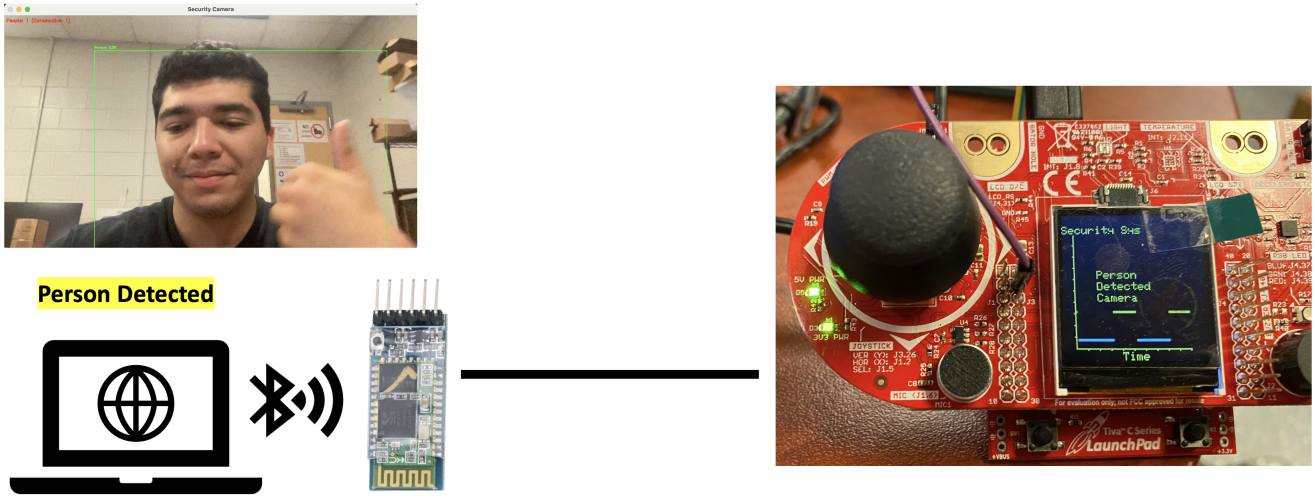

Fig. 2. Sequence diagram of detection → notification → LCD update.

TABLE I
RTOS TASK SET

ID	Period	Function
Task0	1 ms	UART/Bluetooth processing; buzzer chirp on detection
Task1	1 ms	Microphone sampling; RMS level and `heard` flag
Task2	10 ms	LCD rendering; alarm logic
Task3	5 ms	Button debounce; mode toggles

V. HIGH-LEVEL SOFTWARE FLOW

Fig. 2 details the MCU/PC interaction.

VI. DEMONSTRATION PLAN

1) Laptop webcam detects a person; Python script sends ASCII "1".
2) MCU chirps (*Task0*) and paints yellow activity line.
3) Cover camera: MCU clears alarm after 3 consecutive no-detection frames.
4) Switch to microphone mode using Button 1; clap to trigger red line.

5) Hold both buttons for 1 s to arm/disarm and show on-screen status.

VII. Results

- End-to-end notification latency: 85 ± 7 ms (YOLOv4-tiny 55 ms + UART 30 ms).
- System ran continuously for 30 min with no buffer overruns or missed UART bytes.

VIII. Conclusion

The project validates that low-cost MCUs can offload heavyweight vision to a host while maintaining deterministic local control via a minimal RTOS. Bluetooth proved adequate for indoor ranges ($<$10 m) and simplified isolation tests.

IX. Future Work

- Integrate BLE or Wi-Fi for secure pairing and higher throughput.
- Migrate CNN to an edge TPU or MCU-DSP to remove laptop dependency.
- Implement pre-emptive priority scheduler for tighter audio timing.

Appendix A
Full Code Listing

GitHub Key Files: `tm4c/motion_detector.c` and `pc/cv.py`.

Appendix B
Slide Deck

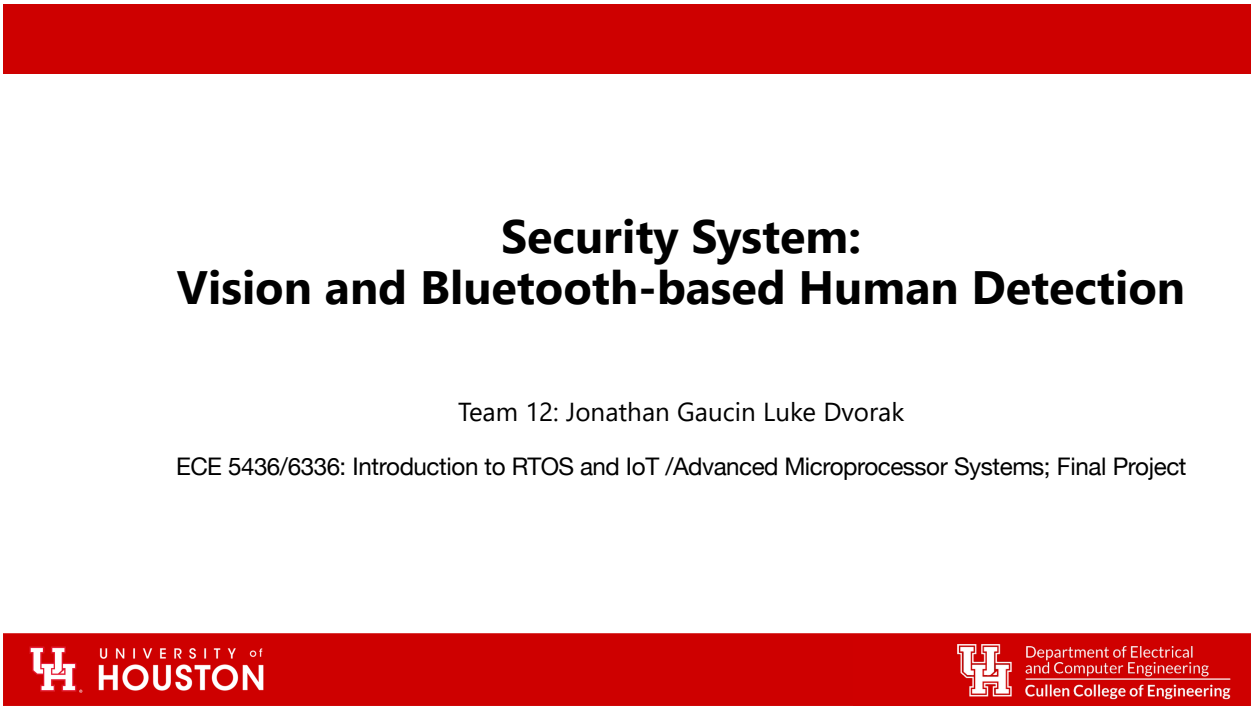

Fig. 3. Title slide (full deck attached separately).