Security System:
Vision and Bluetooth-based Human Detection

Team 12: Jonathan Gaucin Luke Dvorak

ECE 5436/6336: Introduction to RTOS and IoT /Advanced Microprocessor Systems; Final Project

UNIVERSITY of HOUSTON

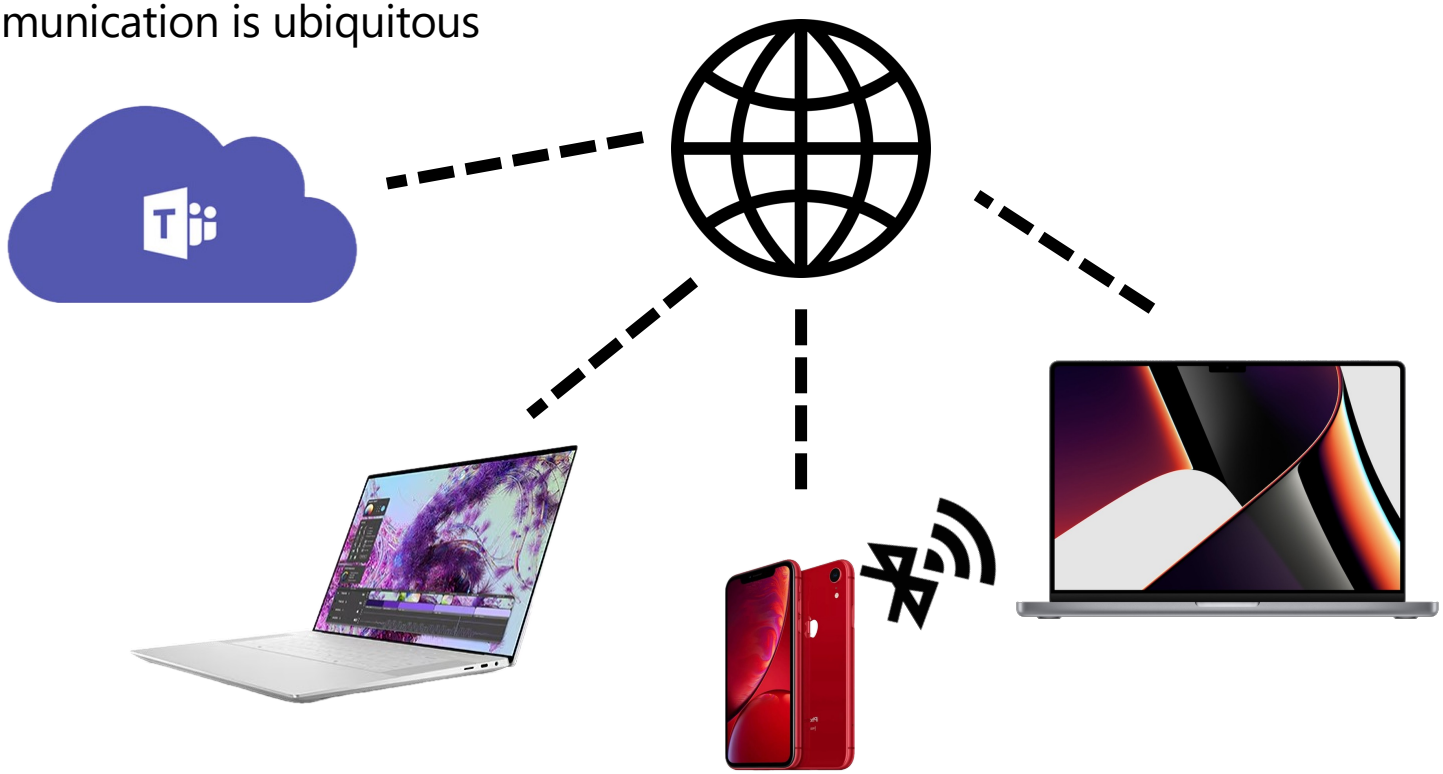
Department of Electrical and Computer Engineering
Cullen College of Engineering

Outline

- Motivation
- Background
  - Computer Vision in Security Systems
- Proposed Solution
  - Hardware
  - Software
  - RTOS
- Results
  - Conclusion
- Future Work

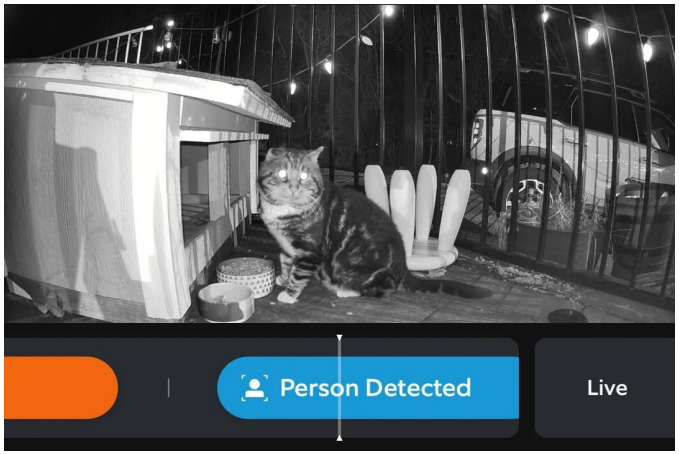
Motivation

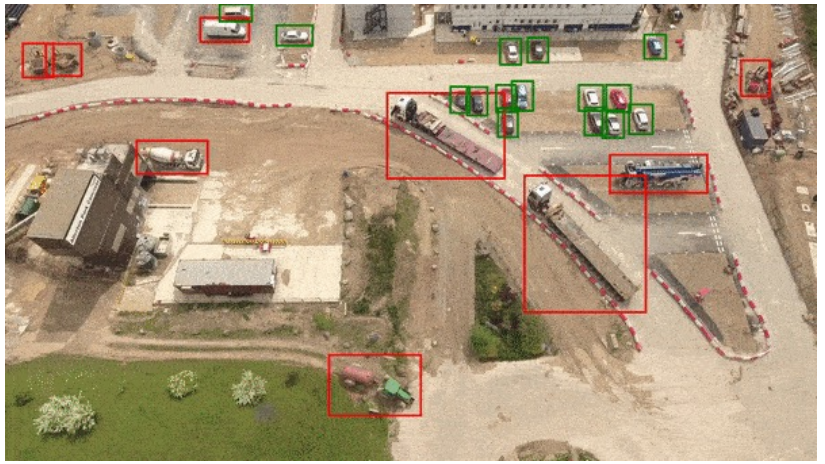
- Wireless communication is ubiquitous



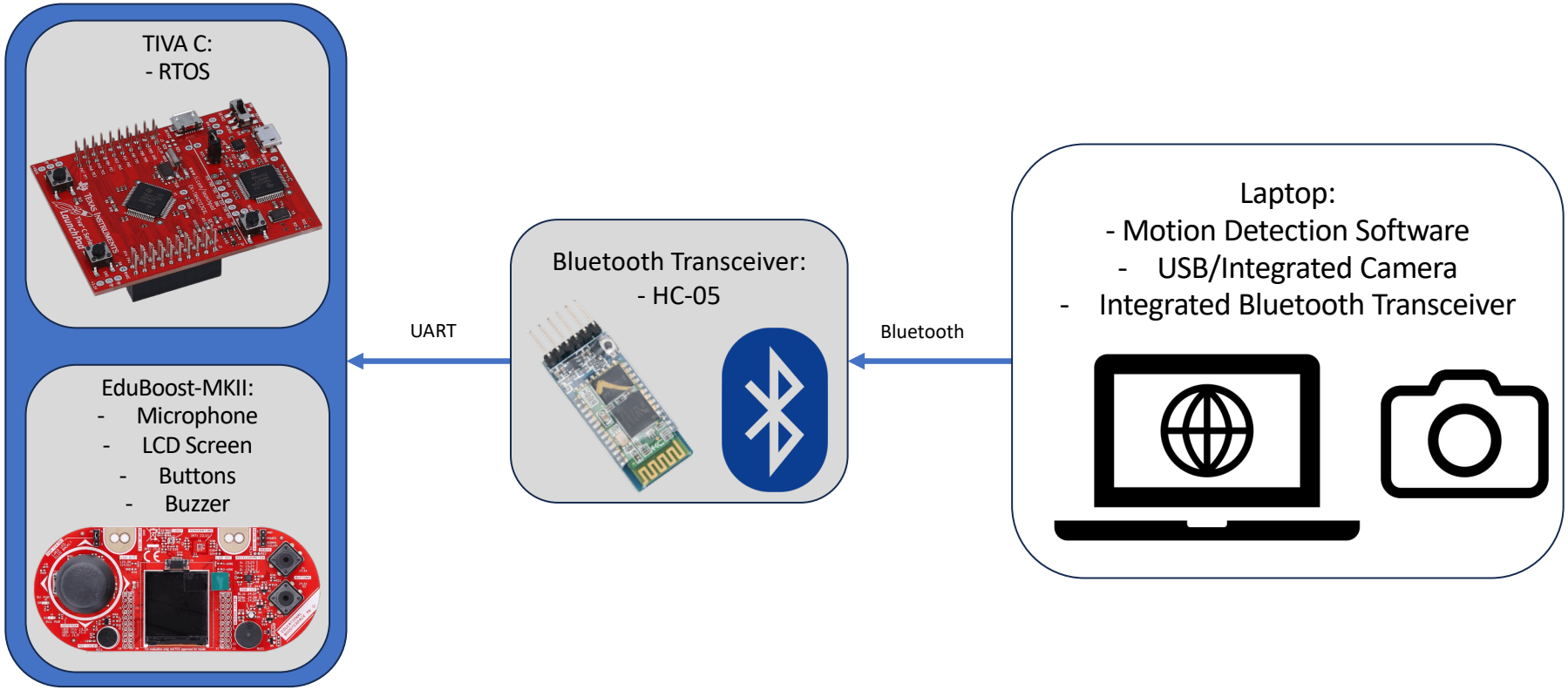
Background

- Computer Vision-based Security Systems

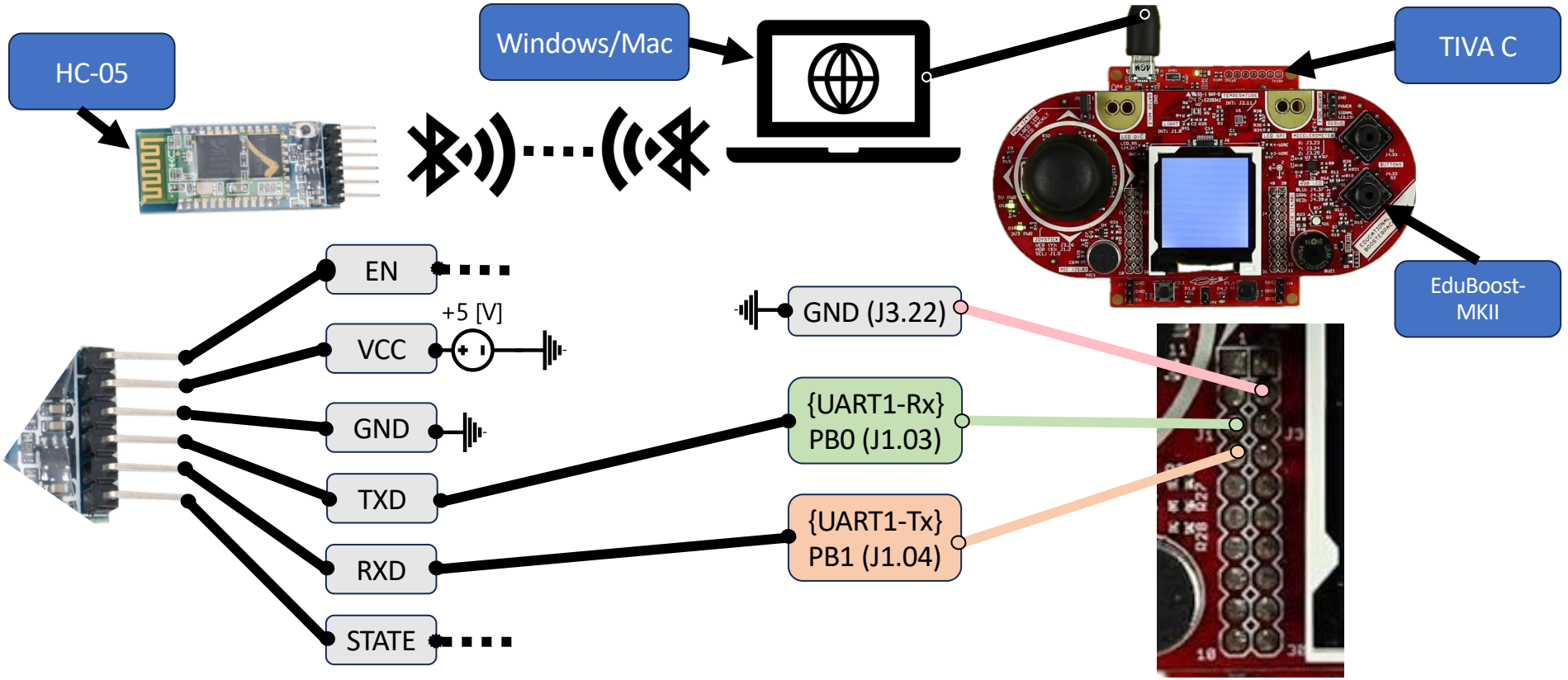




Proposed Solution

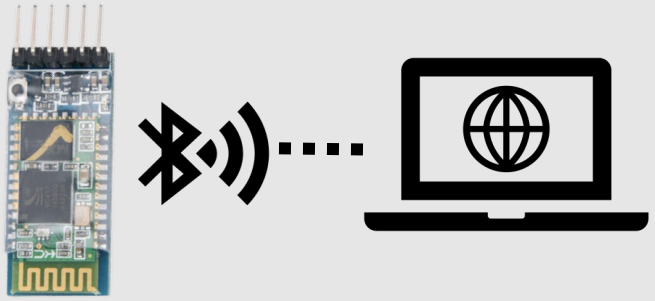


Proposed Solution: Hardware

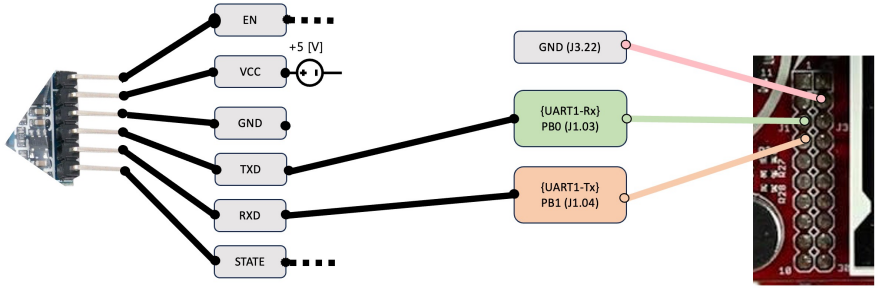


Proposed Solution: Hardware

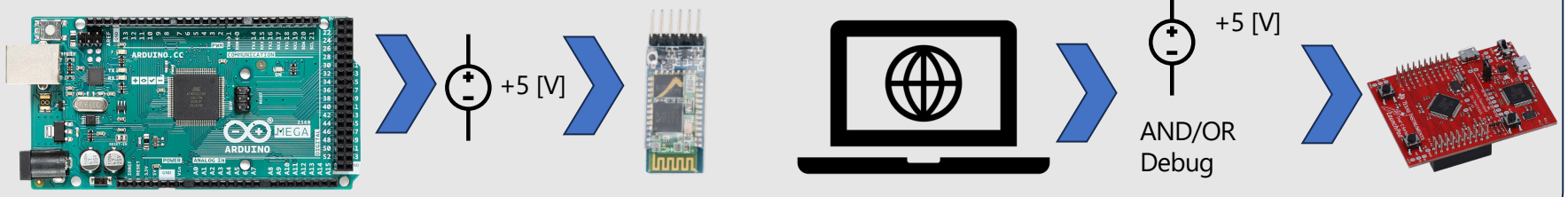
Bluetooth



UART

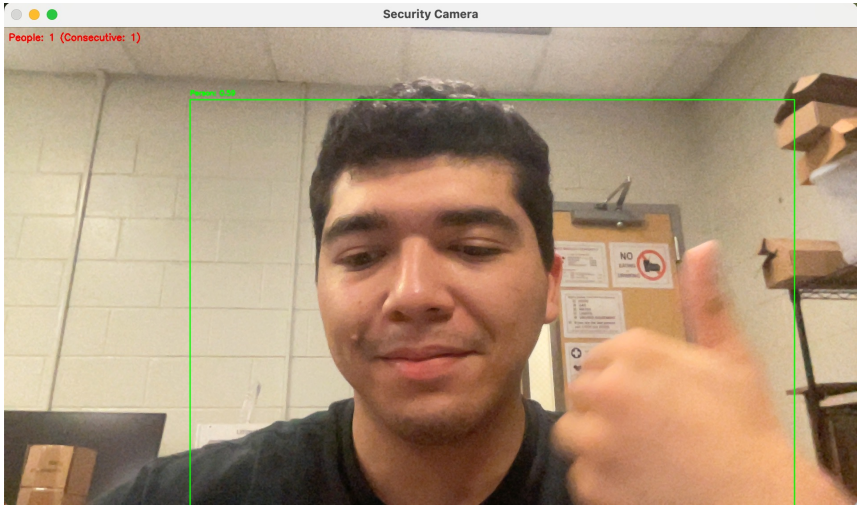


Serial



Proposed Solution: Software

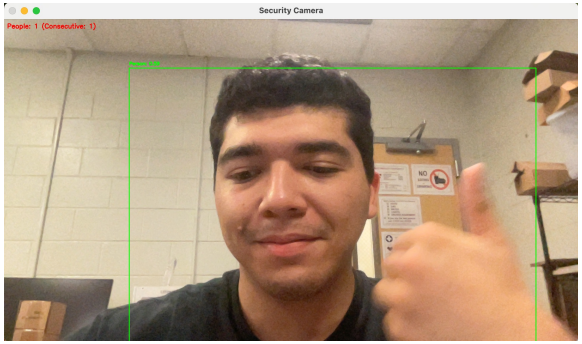
- Python Script (Laptop)
  - Support for Windows and Mac
  - Pyserial for Bluetooth communication
  - YOLO-v4 for Realtime Detection
    - Convolutional Neural Network Model (CNN)



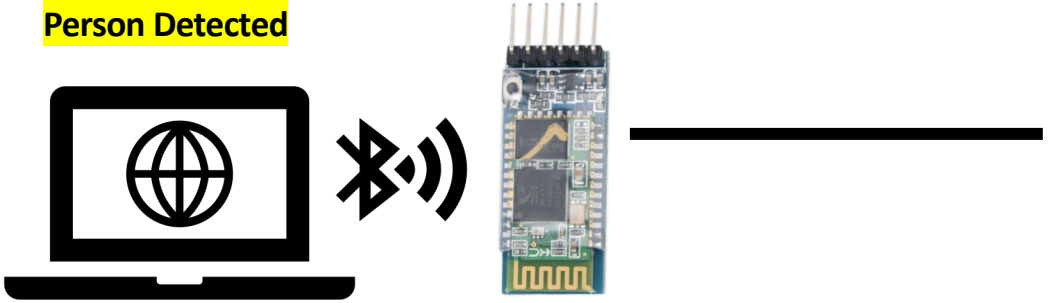

```
print("Starting detection. Press 'q' to quit...")
print("(Detection may take a moment to process each frame)")

# For displaying received messages
font = cv2.FONT_HERSHEY_SIMPLEX
message_history = []
max_messages = 5  # Maximum number of messages to display

while True:
    # Read frame from webcam
    ret, frame = cap.read()
    if not ret:
        print("Failed to grab frame. Check camera.")
        time.sleep(0.5)
        continue

    # Get frame dimensions
    height, width, channels = frame.shape

    # Create a blob and pass it through the network
    blob = cv2.dnn.blobFromImage(frame, 1/255.0, (416, 416), swapRB=True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    # Initialize lists for detected objects
```

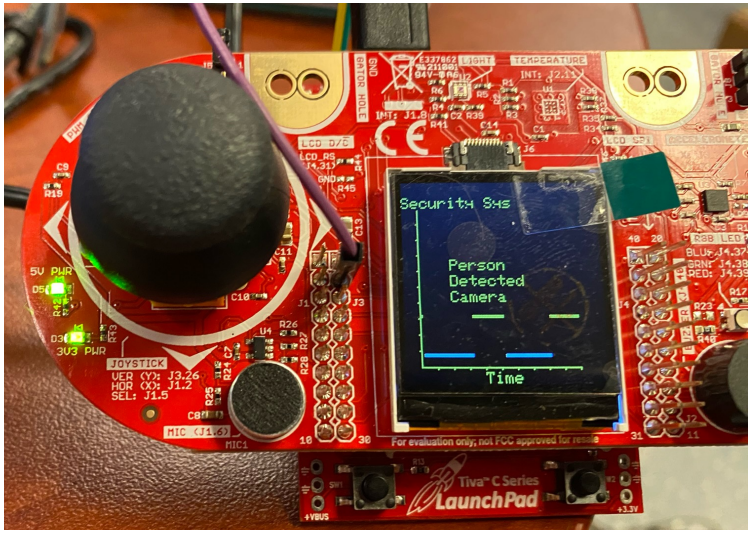
Proposed Solution: Software

- Computer Vision-based Security Systems



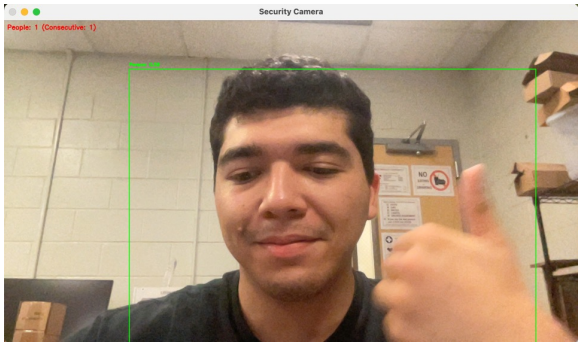
Person Detected



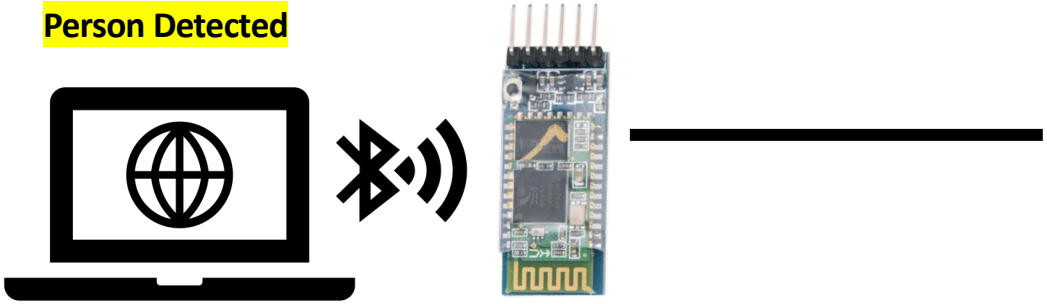


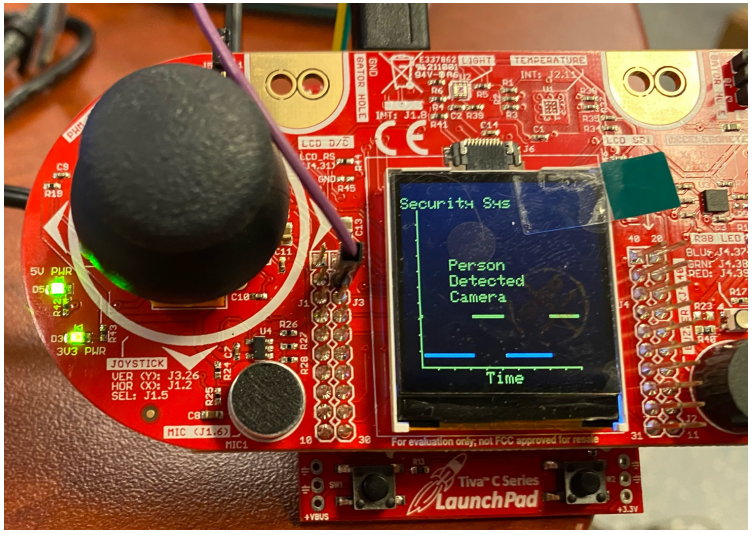
Proposed Solution: Software

- Computer Vision-based Security Systems



Person Detected





Proposed Solution: Software

- TivaC Software Overview
  - Consists of 4 Tasks and 1 Main function
    - Task 0 Handles UART Bluetooth communication
    - Task 1 Handles Microphone Input
    - Task 2 Populates LCD based on global variables
    - Task 3 Monitors Button inputs.
  - These task are performed using a Round-Robin Scheduler
  - The Main function initializes Tasks and their dependencies.
    - This includes Task 1, UART, and LCD Initializatoin

```
380
381  //***************** Main Function *************************
382
383 int main(void){
384    OS_Init();
385    Profile_Init();  // initialize the 7 hardware profiling pins
386    Task1_Init();    // accelerometer init
387
388    //LCD Initializaton
389    BSP_LCD_Init();
390    BSP_LCD_FillScreen(BSP_LCD_Color565(0, 0, 0));
391    BSP_LCD_Drawaxes(PERSONCOLOR, BGCOLOR, "Time", "", 0,"", 0, 500, 0);
392    BSP_LCD_DrawString(0,1, "Security Sys",  PERSONCOLOR);
393    Time = 0;
394    OS_InitSemaphore(&NewData, 0);  // 0 means no data
395    OS_InitSemaphore(&LCDmutex, 1); // 1 means free
396    OS_FIFO_Init();
397
398    //Add threads
399    OS_AddThreads(&Task2, &Task3, &Task0,&Task1);
400
401      // Initialize UART1 at 9600 baud (no PLL, using 16MHz clock)
402      UART1_Init();
403
404      // Send startup message
405      UART1_SendString("\r\nTIVA C Command Processor Ready\r\n");
406
407    OS_Launch(BSP_Clock_GetFreq()/THREADFREQ); // doesn't return, interrupts enabled in here
408
409
410    return 0;            // this never executes
411 }
412
```

Proposed Solution: Software

- Task 0
  - Initialize counts for Buzzer and Detection
  - Receive data from UART FIFO
    - If received echo for Feedback
    - If '1' received set Saw and Chirp
    - Chirp buzzer
    - Increment counts

```
121
122 void Task0(void){
123   uint16_t DetCount = 0;
124   uint16_t ChirpCount = 0;
125
126   while(1) {
127
128     // Stores data from UART FIFO inot int c
129       int c = UART1_ReceiveChar();
130
131       if(c != -1) {  // If data received
132           // Echo character back for terminal feedback
133           UART1_SendChar((uint8_t)c);
134
135       // if c is "1", set Saw to 1 and Chip speaker. additional chirps taken in intervals.
136         if ( c == '1'){
137           DetCount = 0;
138           if (PlotState == Cam && (Saw == 0 || ChirpCount >= 1) && Armed == On){
139             BSP_Buzzer_Set(512);
140             ChirpCount = 0;
141             BSP_Delay1ms(100);
142             BSP_Buzzer_Set(0);
143           }
144
145           Saw = 1;}
146         // count to hold detection for about a second.
147         if( DetCount > 3)
148           Saw = 0;
149
150         DetCount ++;
151         ChirpCount ++;
152
153       }
154   }
```

Proposed Solution: Software

- Task 1
  - This Task Manages the Microphone input.
  - If the plotstate is set to Microphone, the system will take in 10 audio samples to determine if a sound was heard.
    - When a sound is heard, a Heard flag is enabled, and a counter is used to maintain the flags enabled state for a short period of time for Task 2 to read.
    - After this time is passed, the flag is disabled and a new set of sound samples are taken.

```
166  // *********Task1: Takes in shound data and determines if sound detection occur
167   #define SOUND_THRESHOLD 517  // Adjused this based on testing with microphone
168  void Task1(void){
169      static int32_t soundSum = 0;
170      static int time = 0; // units of microphone sampling rate
171      static int soundCounter = 0;  // For debouncing sound detection
172    uint16_t HDetCount = 0;
173      while(1){
174          Profile_Toggle0();
175          BSP_Microphone_Input(&SoundData);
176          soundSum = soundSum + (int32_t)SoundData;
177          SoundArray[time] = SoundData;
178          time = time + 1;
179
180          // Sound threshold detection
181          if(PlotState == Microphone) {
182              // Check if sound is above threshold
183              if(SoundData > SOUND_THRESHOLD) {
184                  soundCounter++;
185                  if(soundCounter > 10) {  // Require multiple samples above thr
186
187                    HDetCount = 0;
188                      Heard = 1;
189                  }
190              } else {
191                  soundCounter = 0;
192                  if(Heard && HDetCount > 20) {
193                      Heard = 0;
194                  }
195              }
196              if(Heard){
197              HDetCount++;
198                BSP_Delay1ms(27);
199              }
200          }
```

Proposed Solution: Software

- Task 2
  - Task 2 takes the global variables of the RTOS and determines what must be displayed on the LCD screen.
  - The first if statement that the task checks determines if both buttons are currently pressed.
    - If they are both pressed, the state of the system will change form Armed to Disarmed, followed by 3 chirps, or Disarmed to Armed, followed by 2 chirps.

```
205  //---------------- Task2 Populates LCD Screen based on g
206
207  void Task2(void){
208    while(1){
209
210  // if both buttons pressed, Arm or disarm the system.
211      OS_Wait(&LCDmutex);
212      if (But2 == 0 && But1 == 0){
213        // button hold time to activate buttons.
214        if(ArmCount >= 10){
215          // if system is on, disarm, and chirp 3 times
216          if(Armed == On)
217            Armed = Off;
218
219            BSP_Delaylms(250);
220
221          BSP_Buzzer_Set(512);
222                BSP_Delaylms(100);
223                BSP_Buzzer_Set(0);
224          BSP_Buzzer_Set(512);
225                BSP_Delaylms(100);
226                BSP_Buzzer_Set(0);
227          BSP_Buzzer_Set(512);
228                BSP_Delaylms(100);
229                BSP_Buzzer_Set(0);
230
231          ArmCount = 0;
232
233          BSP_Delaylms(250);
234      // If system is off, arm and chirp twice.
235        }else if (Armed == Off){
236            Armed = On;
237
238                  BSP_Delaylms(250);
239
```

Proposed Solution: Software

- Task 2 (Cont.)
  - The second if else statement checks if the system's armed state is set to off.
    - If the state is off, it will draw a white line at the bottom of the LCD and notify the user the System is disabled.
  - The Third if else statement checks if the system is in Camera mode, along with if motion has been detected or the test button has been pressed.
    - If motion is detected, a yellow elevated activity line will be made and the user notified of movement.
    - An audible chirp is also made when a person is first detected, and continues in intervals until they are no longer in frame.

```
247         BSP_Delay1ms(250);
248
249         ArmCount = 0;
250       }
251         ArmCount++;
252       }
253     // if no buttons pressed and System is Off, draw white line at bottom of s
254     else if(Armed == Off){
255       BSP_LCD_PlotPoint(30, NOCOLOR);
256       if (ArmWas == On){
257       BSP_LCD_FillRect(25, 40, 90, 40, LCD_BLACK);
258       ArmWas = Off;
259       }
260
261       BSP_LCD_DrawString(5, 5, "Alarm",  NOCOLOR);
262       BSP_LCD_DrawString(5, 6, "System",  NOCOLOR);
263       BSP_LCD_DrawString(5, 7, "Disabled",  NOCOLOR);
264
265     }
266   // if plot state is Cam and either Button 2 is pressed or Saw is 1 indicatir
267     // elevated yellow line and display "Person Detected Camera"
268     else if ( PlotState == Cam && ( But2 == 0 || Saw ==1)){
269
270       if (ArmWas == Off){
271       BSP_LCD_FillRect(25, 40, 90, 40, LCD_BLACK);
272       ArmWas = On;
273       }
274
275       BSP_LCD_PlotPoint(150, PERSONCOLOR);
276
277       BSP_LCD_DrawString(5, 5, "Person",  PERSONCOLOR);
278       BSP_LCD_DrawString(5, 6, "Detected",  PERSONCOLOR);
279       BSP_LCD_DrawString(5, 7, "Camera",  PERSONCOLOR);
280
281       BSP_Delay1ms(25);
```

Proposed Solution: Software

- Task 2 (Cont.)
  - The next else if Statements determine what is displayed on the LCD when either Sound is detected or the test button is pressed during microphone mode.
    - This consists of a elevated red line showing actvitiy, as well as a plot of the sound located below the activity line.
  - The final else if statement determins the action for the Armed system with no input.
    - This resutls in a white line being drawn at the bottom of the screen.

```
280
281         BSP_Delaylms(25);
282
283       }
284       // if plot state is Microphone and either Button 2 is pressed or Heard is 1 indicat
285       // elevated red line and display "Person Detected Mic"
286       else if(PlotState == Microphone && (But2 == 0 || Heard == 1)){
287
288         if (ArmWas == Off){
289         BSP_LCD_FillRect(25, 40, 90, 40, LCD_BLACK);
290         ArmWas = On;
291         }
292
293         BSP_LCD_PlotPoint(SoundData/10, SOUNDCOLOR);
294
295         BSP_LCD_PlotPoint(150, HEARDCOLOR);
296
297         BSP_LCD_DrawString(5, 5, "Person",  HEARDCOLOR);
298         BSP_LCD_DrawString(5, 6, "Detected",  HEARDCOLOR);
299         BSP_LCD_DrawString(5, 7, "Mic",  HEARDCOLOR);
300
301         BSP_Delaylms(25);
302       }
303       //If system is armed but no inputs recived, draw a white line at bottom of screen.
304       else if((Saw == 0 && PlotState == Cam) || But2 != 0  ){
305         BSP_LCD_PlotPoint(30, NOCOLOR);
306
307         BSP_LCD_FillRect(25, 40, 90, 40, LCD_BLACK);
308       }
309       BSP_LCD_PlotIncrement();
310       OS_Signal(&LCDmutex);
311     }
312   }
313
```

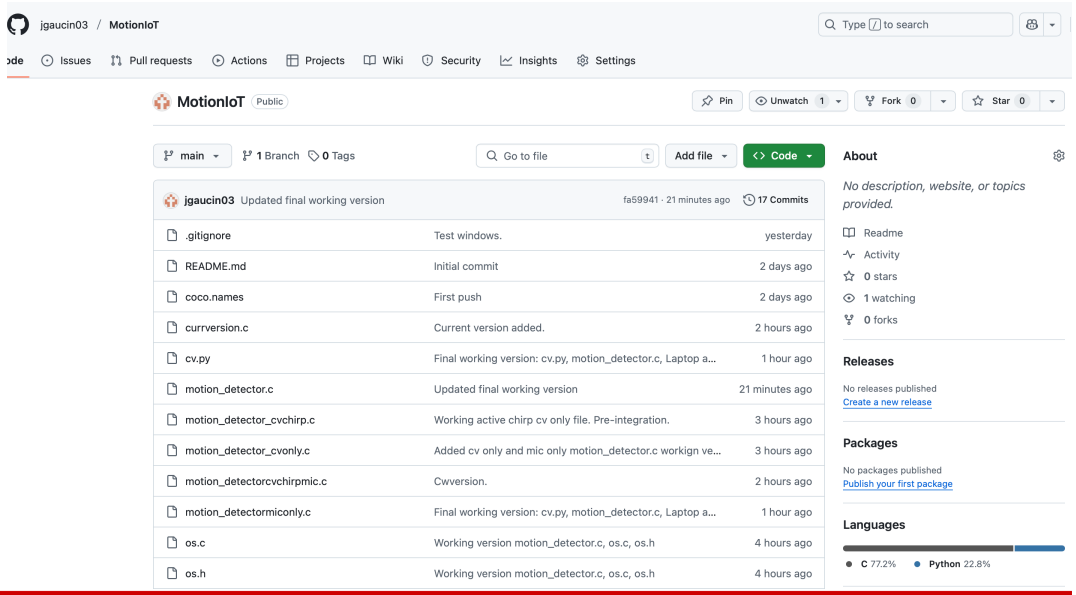
Proposed Solution: Software

- Task 3
  - Task 3 Monitors the State of Buttons 1 & 2.
  - Both buttons emit a chirp when pressed.
  - Button 1 does not perform any additional tasks.
  - Button 2 handles PlotState Switching when pressed.

```
313
314  //------------Task3 Monitors Button Inputs.-------
315
316 void Task3(void){
317    static uint8_t prev1 = 0, prev2 = 0;
318
319    BSP_Button1_Init();
320    BSP_Button2_Init();
321    BSP_Buzzer_Init(0);
322    BSP_RGB_Init(0, 0, 0);
323    while(1){
324      Profile_Toggle3(); // viewed by a real logic analyzer to know Task3 started
325      BSP_Buzzer_Set(0);
326      But2 = BSP_Button2_Input();
327      if((But2 == 0) && (prev1 != 0)){
328
329        BSP_Buzzer_Set(512);           // beep until next call of this task
330      }
331      prev1 = But2;
332      But1 = BSP_Button1_Input();
333      if((But1 == 0) && (prev2 != 0)){
334                // redraw axes on next call of display task
335
336              if(PlotState == Cam){
337                  PlotState = Microphone;
338              }else if(PlotState == Microphone){
339                  PlotState = Cam;}
340
341
342        BSP_Buzzer_Set(512);           // beep until next call of this task
343      }
344      prev2 = But1;
345
346      BSP_Delay1ms(5); // very inefficient, but does debounce the switches
347    }
```

Results

- Demo Presentation
- Code is Open-Source, Available at:
  - https://github.com/jgaucin03/MotionIoT



Conclusion

- Successfully detected a person using:
  - EduBoostMKII Peripherals:
    - Microphone, Button, Buzzer, LCD screen
  - State-of-the-art Detection Methods:
    - Computer Vision
  - Internet of Things
    - Fog Computing
    - Bluetooth Communication
  - RTOS