Ben Roy – Summer 2021 Projects

# Hardware Trigger Subsystem

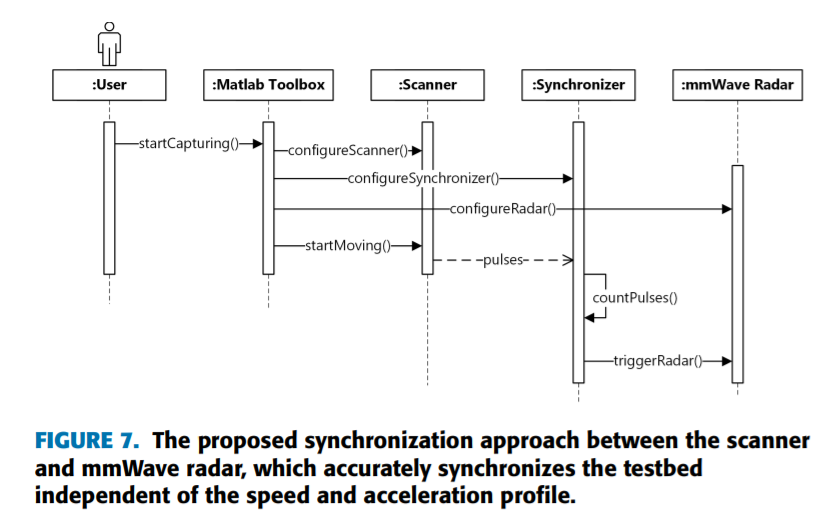
* Requirements:
  + Develop real-time software to trigger the radar at desired times
  + Control input from different devices
    - Count pulses from stepper driver



* + - Input from distance sensor (lidar, laser, closed loop motor, IMU)
  + Inputs:
    - Desired distance between triggers from PC over UART
    - Horizontal scan length from PC over UART (for XY scanner application)
    - Pulse count from PC over UART (for stepper driver application)
    - Control input
  + Outputs:
    - HW trigger (pulse) to radar at desired instance
  + Hardware:
    - ESP32 (might need to buy more)
    - Necessary cables, etc.
    - TI radars (already have)
* Deliverables:
  + ESP32 source code
  + Documentation on ESP32 source code
  + MATLAB scripts
  + Documentation on MATLAB scripts
* Tasks:
  + Learning the basics of the ESP API – [main page](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/)
    - Develop code for a simple pulse train following TI radar constraints
      * See mmwavelink/src/rl\_sensor.c around line 976 for requirements
        + Pulse width should be between 25 ns and 1 us
      * [Here](https://e2e.ti.com/support/sensors-group/sensors/f/sensors-forum/972613/iwr6843isk-hardware-trigger) says rise time needs to be ~3 us
    - Use these libraries
      * [GPIO](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html)
      * [UART](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/uart.html) (USB communication to PC)
    - Create short manual on compiling the ESP code
    - Inputs to ESP32 from MATLAB:
      * Pulse width and period from PC over UART
    - Outputs from ESP32:
      * Pulse train with expected characteristics and rise-time < 1 us
    - Deliverables:
      * ESP32 source code for pulse train software
        + Well commented for future use
      * MATLAB software for sending commands to ESP32 over serial
        + Don’t worry, this is easy
      * Documentation on how to use the software: both how MATLAB sends commands and how ESP32 receives them etc.
      * Separate documentation on how to compile ESP32 software
  + Create software for HW trigger on XY scanner (synchronizer)
    - Stepper driver outputs pulses to the motor
    - Monitor these pulses with the ESP32 and count them
    - Knowing the conversion between pulses and distance, trigger the radar every time a certain number of pulses have been sent by the stepper driver
    - Implement for the back-and-forth scanning pattern
      * Before we start the scan, the ESP32 will plan the scan based on the distance between the triggers and the total horizontal scan length . This plan will consist of exact locations where the radar will be triggered. For example, we determine that the distances will be , and there will be some excess distance in the first direction of scanning which will need to be compensated for when scanning in the opposite direction. See the diagram below.
      * ESP32 will track distance up until it gets to the horizontal scan length, then it will “back track” and capture at the exact same locations this time while the platform is moving in the opposite directions
      * Needs to account for when the total distance is not an integer multiple of the distance between triggers



* + - Use these libraries
      * Others from above
      * [Pulse counter](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/pcnt.html)
    - Inputs to ESP32:
      * Desired distance between triggers:
      * Horizontal scan size:
      * (Maybe the number of triggers/pulses per horizontal scan?) N
      * Pulses from stepper driver
    - Outputs from ESP32:
      * HW triggers to the radar
      * Acknowledgement to the PC over UART that the command has been received
        + setup 2 500 64
    - Deliverables:
      * ESP32 source code
        + Well commented for future use
      * MATLAB software for sending commands to ESP32, verifying that ESP32 received the command (reading back acknowledgement)
      * Documentation on how to use the software: how MATLAB sends the commands, receives the acknowledgement, and how the ESP32 receives the commands, plans the scan, sends the acknowledgement, reads in the pulses from the stepper driver, and sends the HW trigger
      * Tested with the scanner setup and working
    - Provided:
      * Josiah has made a MATLAB script for controlling the XY scanner with relevant places for you to add your code in MATLAB and test the HW trigger
      * For more info, see [Muhammet’s Paper](https://ieeexplore-ieee-org.libproxy.utdallas.edu/document/9136646)
        + Below is a helpful graphic on the control flow of the program
        + He has already worked on this software, but since it was in joint with TI, we do not have access to it as it’s now under NDA with TI
        + [Here](https://github.com/meminyanik/MIMO-SAR-mmWave-Imaging-Toolbox/tree/master/SAR_GUI_xWR1xxx_AMC4030/SARSYNC) is a link to what is available from that code
        + As you are working on your code, you can develop a similar diagram



* + Create software for HW trigger on XY scanner with dual-radar
    - Create another set of source code starting with what you did for the single radar case but this time it accommodates two radars
    - Assume we have two radars that are separated by some distance
    - Again, we will count the pulses from the stepper driver just like before
    - The ESP32 will also plan the exact locations again, like it did for the single radar case; however, when it triggers, it will need to take into account the two different positions
    - Now, the plan will consist of locations for triggering radar 1 and the locations for triggering radar 2 knowing that it is horizontally offset
    - The ESP32 will trigger each radar when it is at the correct position
      * For example, let’s say that we are monitoring the pulses and estimate the position to be
      * Let assume radar 1 is located at  and radar 2 is located at  , meaning is horizontal position is behind
      * Take this into account when triggering so that the radars are triggered at exactly the same horizontal location
    - Inputs to ESP32:
      * Desired distance between triggers:
      * Horizontal scan size:
      * Distance between the radars:
      * (Maybe the number of triggers/pulses per horizontal scan?)
      * Pulses from stepper driver
    - Outputs from the ESP32:
      * HW triggers to radar
      * Acknowledgement to the PC over UART that the command has been received
    - Deliverables:
      * ESP32 source code
        + Well commented for future use
      * MATLAB software for sending commands to ESP32, verifying that ESP32 received the command (reading back acknowledgement)
      * Documentation on how to use the software: how MATLAB sends the commands, receives the acknowledgement, and how the ESP32 receives the commands, plans the scan, sends the acknowledgement, reads in the pulses from the stepper driver, and sends the HW trigger to each radar
      * Tested and working with the dual radar setup to trigger the radar when/where we want it to
    - Provided:
      * Use the code that you develop for the single radar case
      * Josiah has made a MATLAB script for controlling the XY scanner with relevant places for you to add your code in MATLAB and test the HW trigger for the dual radar case
  + Create software for HW trigger with different control input
    - Create another set of source code that essentially modifies only the control input dictating the location estimation (where is the radar currently) aka
    - Different options:
      * Closed loop stepper motor?
      * IMU/position sensor
      * Laser or lidar highly precise location estimation
      * Work with Jayson as this may overlap with his work with Microsoft
      * Figure out how these sensors send the position information and track it with the ESP32 to trigger in the same way
    - All you need to change is how the location is tracked, so this can be easily implemented for both single or dual radar
    - Inputs to ESP32:
      * Same as single/dual radar
      * Control input from new position sensor
    - Outputs from ESP32:
      * Same as single/dual radar
    - Deliverables
      * ESP32 source code
        + Well commented for future use
      * MATLAB software for sending commands to ESP32, verifying that ESP32 received the command (reading back acknowledgement)
      * Documentation on how to use the software: how MATLAB sends the commands, receives the acknowledgement, and how the ESP32 receives the commands, plans the scan, sends the acknowledgement, reads in the pulses from the stepper driver, and sends the HW trigger to each radar
      * Tested and working with whatever sensor type you choose
    - Provided:
      * Use the code that you already developed
      * You can test it in the same was the other tasks

# Important notes:

* Keep track of EVERYTHING
  + Every day you work, keep a diary (use OneNote or Word) of what you are trying and what works/doesn’t work with screenshots (+Shift+s [is an easy shortcut that lets you select where to screenshot and puts it onto your clipboard to paste anywhere you want](https://www.nextofwindows.com/windows-shift-s-new-way-to-take-screenshots-windows-10-creators-update)), links, etc.
    - This is most important if you are debugging someone else’s code or learning the API. Keep track of links that were helpful in a diary or things that were recommended by various people. Also keep track of things that you notice that aren’t working properly or are finicky
  + Remember the goal is always that someone could come behind you and repeat exactly what you did, but with ease
  + Use GitHub and a private repository for each task
  + As you are developing code, document it well with comments in the code, function descriptions, and a “user manual” for understanding the various functionalities