Project Roadmap – Radar-Room-Scanner

Organizational

- General information: ELO: MEM/Project Thesis MEM | E-Learning
- The project thesis has a workload of 180 h (6 ECTS).
- "Project thesis is work at the level of an engineer on a research or development project."
    - That means it is independent and self-reliant work!
    - Finding solutions based on literature on your own and present and discuss them.
- Deadline for the project thesis will be the end of the semester, at the latest 30.09.2025
- Assignment:
    - An OTH Gitlab repository Projects · GitLab (add us to the GIT) with
        - Source code & supportive scripts
        - Dependent on the project and applicability:
            - CMake / build instructions for MCUs
            - Flashing instructions & dependencies for required tool chains (e.g. STM32CubeIDE, ModusToolbox, CCS, …)
            - Well-structured code with explanations (e.g. Jupyter Notebook(s))
            - Modular hardware abstraction layer (HAL)
        - README with setup instructions, requirements, …
        - Concise documentation (avoid unnecessary bloat) for reproduction
        - All relevant files for reproducibility
        - Exported logs / plots / evaluation results as reference output
    - Final live presentation & demo
    - **Reproducibility**
        - Project must be reproducible on a new PC & understandable using only the GitLab instructions
        - Provide tested setup script / Dockerfile if environment is complex
    - The project must be able to set up on a new PC solely with the instructions of the Gitlab & also be understandable with the assisting documentation and explanations
    - Recurring meeting content
        - A very short PowerPoint presentation

        - Present work done, findings, current status, next steps
        - Open, self-critical discussion & planning

0 Good Practice

- Document every **important step** and **parameter**
    - Directly from the beginning – not retroactively!
    - For reproducibility and reconstruction: ensure future students or collaborators can set up and understand the project from scratch
        - Without this, the long-term value of the work is significantly reduced
        - Including things that did not work, too – they provide valuable insights
    - If suitable: justify methods and implementation choices with literature or sound reasoning
- Clean code
    - Maintain modularity and well-commented code
    - Version control everything using Git
    - Use structured naming & folder conventions
    - Benchmark and log systematically
- Be self-critical and transparent -> Quality over Quantity

1 Theory

This chapter establishes the theoretical groundwork required to interpret the sensor data and deploy the necessary algorithms.

- Radar fundamentals
    - FMCW Radar
        - Working principle
        - Chirp design
    - Signal processing
        - Bandpass filtering
        - FFT (range/speed)
        - Peak/CFAR detection
        - Angle / clustering
        - Mapping layer
- IMU & Magnet sensor fundamentals
    - Attitude Estimation Theory
    - Possible Sensor fusion algorithms
    - Calibration methodologies
- MCU architecture
    - CY8CKIT-062S2-AI
        - FMCW sensor

        - 6-axis IMU
        - 3-axis Magnetic sensor
    - Toolchain
        - ModusToolbox, … (Processing is done in Python on PC!)
- Mapping & Localisation Concepts
    - E.g. Occupancy-grid mapping
    - E.g. Scan-matching / ICP
    - mmWave-SLAM research (waveSLAM)

2 FMCW Radar & Mapping

This section covers the radar sensing and mapping core of the project. Using the onboard BGT60TR13C FMCW radar sensor to acquire distance measurements and build a map of the room. Choosing a suiting chirp configuration for this task is crucial for extracting optimal range estimations for error reduction. Multiple mapping approaches should be researched (simple point maps to SLAM), as well as the possibilities for 2D, 2.5D or 3D maps with FMCW radar for room-scanner.

- Setup the Hardware and familiarize with radar fundamentals and sensor capabilities
    - Raw data acquisition and streaming to PC (python)
    - Hardware configurations for optimal measurements
        - Document setting tradeoffs in e.g. an ablation table in the git
- Signal processing pipeline for range detection
    - FFT, Frequency bins, …
    - Noise floor estimations / thresholding for object detection
    - CFAR
    - Algorithm for multiple peak handling (multiple objects)
    - Focus first on stationary case for repeatability of measurements and testing range measurement accuracy against known distances
        - Document the experiment settings and parameters thoroughly
- Mapping from polar data
    - Accumulate measurements into spatial map
        - Again, first as stationary/only rotating scanner (to reduce complexity)
    - Convert polar coordinates (rotated angle and distance of measured object(s)) into Cartesian points
        - Result will be a rudimentary 2D map -> evaluate radar's sensing capability by doing this
- Literature review for mapping approaches for range or even radar sensors
    - E.g. Occupandy grid map
    - E.g. Point cloud mapping

    - E.g. Feature based SLAM
    - E.g. Scan matching
    - In studies like
        - mmWave radar SLAM
    - Summaries pros/cons (again in ablation table in git)
    - Include References to studies!
- Based on your findings, choose the best suiting choice
    - It is okay to initially select a simpler approach; full SLAM could be a bit to excessive
    - Justify with clear rationale
- Implementation and testing
    - How much time does a full 360° scan take? (Again, probably still 2D)
    - Test the mapping in a room with visualization
- Benchmark and refine
    - Smallest trackable object
    - Max distance
    - Repeatability of the map
    - …

3 IMU-Based Stabilization

This section addresses the challenges of using the device in a handled (moving/unsteady) mode. Deviations in the 3rd dimension can distort the mapping. The goal is to use the IMU and magnetometer (6+3 axis) to estimate orientation (roll, pitch, yaw) in real time. By having this information, we can compensate the device motions. Implementation of sensor fusion for stable sensor readings and integration into the mapping pipeline will be key. Multiple algorithms should be considered (complementary filters, Kalman filters, …) and as always justified. This will enable the transition from stationary rotating scanner to handheld scanner, while containing accuracy.

- Orientation estimation via sensor fusion (3D Gyroscope, accelerometer and magnetometer) for device's orientation in space (quaternion/Euler angles – yaw, pitch, roll)
    - Interface with sensors (PC and python)
    - Literature research common algorithms for IMU fusion
        - Focus on IMU-based stabilization and mapping
            - E.g. Of mobile scanning devices and how they perform AR mapping
            - E.g. How drone stabilization approaches could be relevant for handheld scanners
            - Focus on orientation estimation
        - General fusion methodologies

                - E.g. gradient-descent based attitude and heading reference systems (AHRS)
                    - E.g Madgwick's filter
                    - E.g. Mahonys's filter
                - E.g. More computational intense filters
                    - E.g. Kalman filter
                    - E.g. extended Kalman filter
                - E.g. Complementary filters (maybe simpler) for drift correction
                    - Gyro integration (=high-frequency orientation changes) blended with accelerometer/magnetometer (=low-frequency absolute reference)
            - Review literature and possible available open-source implementations
            - Compare pros/cons (ablation table in git)
- Sensor calibration and alignment
    - Think of fitting calibration methodology and algorithms for stable baseline
    - Offset bias measurement
    - Define radar orientation axis mapping
        - E.g. radars forward facing direction -> X-axises
        - E.g. radar is facing east -> yaw=0
        - E.g. radar level horizontally -> pitch=0
- Real-time orientation filtering
    - Based on your findings implement the chosen sensor fusion algorithm(s) for testing
    - Thorough testing of performance
        - Does tilting the device result in correct changes?
        - When kept still is it drifting?
        - Outdoor vs indoor (magnetic field of electricity lines)
- Stabilize mapping
    - Integration of the orientation data into the radar mapping pipeline
    - For each radar reading:
        - Retrieve device pitch, roll and yaw from the IMU-fusion
        - For each radar reading:
        - Pitch/Roll correction:
            - Compute true horizontal distance
        - Yaw correction:
            - Use IMU-derived yaw as the scan angle instead of manual inputs
    - Implement in phase:

        - Phase 1: Support freehand yaw rotation while keeping device level
        - Phase 2: Add pitch/roll-based vertical angle compensation
    - Test and validate
        - Perform controlled hand-held sweep scans
        - Compare resulting map to earlier static-axis scan map
- Performance and corrections
    - After integration, evaluate how well stabilization works (think of fitting experimental setting)
        - Baseline scan (stationary)
        - Handheld scan (stationary)
        - Compare results and analyze discrepancies
            - E.g. timing misalignments between IMU and Radar
            - Quantify orientation error impact
    - Refine filter tuning
    - Document every experimental setting thoroughly
- The device should be able to be rotated freely (and mildly tilted) by a user and still produce a coherent map of the environment!

4 Localization & Integration

This final domain tackles the broad problem of the movement of the device in the room (not just rotation in one spot). How do we keep track of its position and integrate multiple scans in a single map? This can be understood under SLAM (Simultaneous localization and mapping). The goal is to investigate and test various methods for tracking the scanner's position and heading over time. This will be done based on the available sensors and the generated map data. Different options should be assessed (e.g. 9-axis IMU; e.g. radar-based SLAM or external reference points) and evaluated.

- Research localization strategies
    - Indoor localization techniques relevant for the system (for trajectory [x, y and heading] estimation)
        - IMU Dead-Reckoning
            - Double integrating accelerometer data to get position changes and gyroscope for heading changes. (drift will be a problem)
        - Visual/Spatial Anchors
            - If objects (and thus map features of these objects) are dominant enough
        - Radar based localization/SLAM
            - Radar scan matching (distinct features + iterative closest point algorithm)

                - mmWave radar SLAM
                - mmWave GraphSLAM
                - waveSLAM
        - Pro/Con list (ablation table in git)
- Test localization approach(es)
    - Based on your findings, select method (or hybrid)
        - E.g. Hybrid: IMU for relative movements and heading, but periodically correct the drift be recognizing if the radar sensor sees a previously mapped object (thus closing the loop)
        - E.g. Full: Autonomous SLAM
        - E.g. Manually divide mapping into segments
            - User stands in one spot -> scans -> walks to new spot -> scan again -> …
    - Clearly state the assumption of your chosen methodology and its limitations
- Integration of mapping and tracking
    - Combine the localization with the mapping pipeline to build a unified system
    - Choose a fitting origin and axes convention
    - When device moves, transform incoming radar measurements by device orientation (section 3) and current estimated position
        - Managing coordinates and transformations is key
            - Function to convert radar readings (range, angle relative to device) + device pose (position x,y and heading) into global coordinates (X,Y)
        - E.g. When device moves 1m to the right, radar sees a wall 3m ahead, the wall's global position should be plotted 1m right of where it would have been, if device hadn't moved
- Optional: Consider implementing full SLAM
    - The final goal should be creating an own algorithm and pipeline
    - Fallback could be (but not desired) integrating in existing SLAM framework
- System testing and demonstration
    - Test full mapping system in real environment but under experimental settings
    - Think of fitting experimental settings
        - E.g. start in one corner -> scan -> walk to center -> scan -> same/another corner
    - Compare constructed map against known layout
        - Note discrepancies
            - E.g. curved walls
            - E.g. missing objects
            - …

- Evaluate accuracy of localization
  - When ending the measurement at the same spot (loop closure) what does the system think where you are at?
    - Drift / Offset correction
- Evaluate computational performance needed
  - Is it possible to run on RaspberryPi?
  - How long would it take?
  - Real time?