Development Module for Radar Safety Sensor in Single-Track Vehicles

Martin Ťavoda
FEEC Department of Radio Electronics
Brno University of Technology
Brno, Czech Republic
martin.tavoda@vutbr.cz

***Abstract*—This article describes the hardware and software design of a development module for the Radar Safety Sensor (RSS). RSS uses a Frequency-Modulated Continuous Wave (FMCW) radar to track moving objects behind single-track vehicles. The current radar configuration and antenna have significant object tracking deficiency when the vehicle makes a turn and the radar Field of view (FOV) tilts. The assumed solution is to add Inertial Measurement Unit (IMU) data to the tracking algorithm. The IMU is implemented in the designed development module, which also provides an efficient development and debugging environment.**

***Index Terms*—embedded device, cyclo-safety equipment, ROS 2, micro-ROS, FMCW radar**

I. Introduction

The designed module will serve the company ALPS ALPINE Co., Ltd. in the development of Radar Safety Sensor (RSS). Together with the ALPS Generic Radar 5, it should result in a similar device as the already existing Ride Safety System RS 1000 or GARMIN Radar Tail Light. These devices are safety equipment for cyclists that offer information about vehicles behind them. If the vehicle is approaching too quickly or it is driving too close the cyclist and also the driver are warned [1].

The module purpose is to process data from an IMU and send them to a Single Board Computer (SBC) through a Robot Operating system 2 (ROS2) domain. As an output, it collects result data from a ROS2 domain and triggers an alert. Additionally, it provides robust connectors, power management, ANT+ connection, and physical mount to single-track vehicles, specifically a bicycle. The whole system will be supplied from an external 18 V Li-Ion accumulator.

II. Problem formulation

ALPS ALPINE Co., Ltd. is developing a new cyclo product using a proprietary radar system with a new tracking algorithm. However, they ran into a problem. In the default scenario, the radar is mounted under the bicycle seat and the bicycle is moving in a straight line (no tilt). The radar sees multiple points, where some of them are reflections from a static background (the blue dots in Figure 1), and the rest are reflections from actual moving targets behind the bicycle (the green dot in Figure 1).

In this scenario, these two conditions are easily distinguishable in processed radar data output and the ego-motion vector

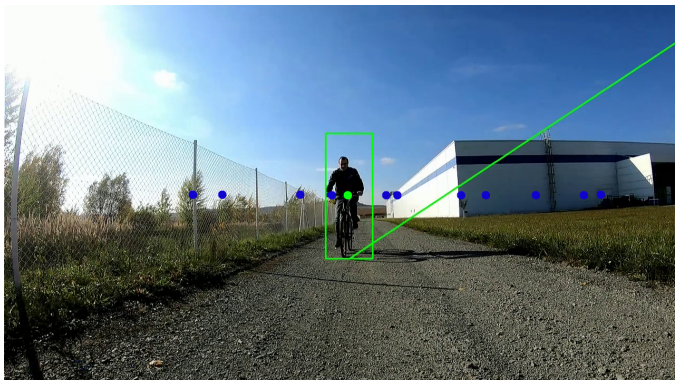

Fig. 1. Correct tracking - straight ride

can be calculated from these reflections of static background. Ego-motion vector describes the motion of an object relative to the rigid scene [2]. In this context, it consists of two elements, direction (forward or backward), and speed (approaching or receding). In the current radar configuration, the radar FOV is only a horizontal line, there is no elevation data, and the ego-motion vector is only one-dimensional. The reflections from moving targets are clustered and classified, and a tracking algorithm is used. From these data, it is possible to estimate the moving target trajectory, and speed (approaching or receding), classify if it's a car or truck, and calculate the possibility of a potential collision.

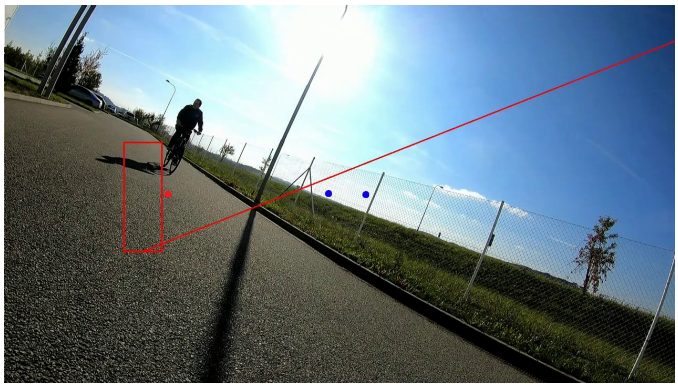

Fig. 2. Tracking degradation - ride in turn

However, if the bicycle tilts, for example, when turning, the radar FOV also tilts with the same angle and this condition cannot be detected from radar data alone. Since the ego-motion vector is only a single dimensional and the height of the radar from the ground changes, the previously tracked object could be tracked incorrectly (see in Figure 2) or can disappear altogether.

A similar problem is described in [3], but the company requested a simpler approach. One possible solution to this problem seemed to be adjusting the tracking algorithm parameters to be more lenient, but this resulted in insufficient accuracy. The development team decided to try to add to the tracker algorithm some external information about radar tilt and position in space to enhance the tracking estimation. This information can be obtained from an IMU located in a development module.

After the successful implementation of IMU data into the tracking algorithm by the ALPS team they will need to easily show the performance of this product. For this purpose, the development module offers output peripherals such as highly luminous Light-Emitting Diodes (LEDs), a loud buzzer, and also in the future the ability to connect to external devices such as wireless ANT+ fitness accessories.

III. System design

The system design is shown in Figure 3. The development module for the RSS consists of multiple blocks. The main part is the microcontroller ESP32 which collects the raw data from the IMU using the I^2C bus. These data are then processed and sent to the ROS2 domain via Ethernet. Ethernet connection is provided by Media access control sublayer (MAC) and Physical layer (PHY) interfaces. MAC is included in ESP32 and PHY is an external component. They communicate with each other by Reduced Media-independent interface (RMII). Ethernet was preferred over Wi-Fi due to its lower latency. As IMU is used the MPU5060 chip and the PHY interface is provided by IP101GR Integrated circuit (IC).

The SBC connected to the ROS2 domain records data from the development module and the radar Printed Circuit Board (PCB). These data are stored in the format of ROS2 database.

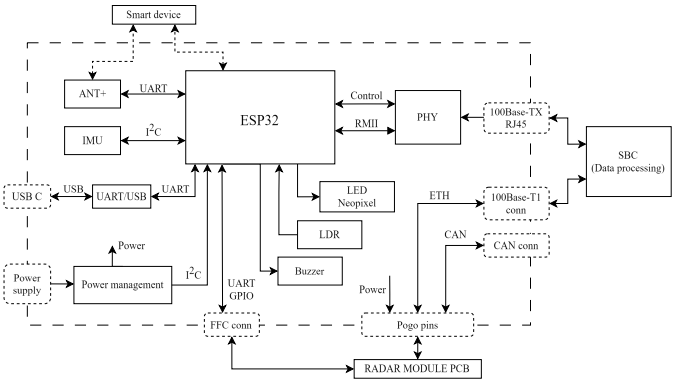

Fig. 3. Development module block diagram

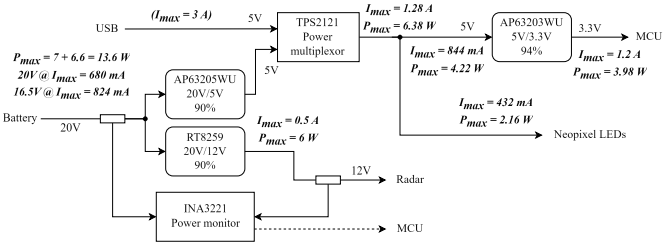

Fig. 4. Power management block diagram

The Alps team can afterward playback these data and use them to train the neural network that currently presents the tracking algorithm. The output (a threat level) of the finished algorithm should be sent back to the ESP32, whose task is to provide an alert by visual or sound peripherals.

As visual output the programmable RGB LEDs WS2812B mini are used. They are placed on the perimeter of the PCB and the brightness of these LEDs is adjusted by the software according to the amount of ambient light. This level is read from the external photoresistor by the Analog-to-digital converter (ADC) included in the ESP32.

The output threat level from the tracking algorithm will be also wirelessly sent to an external smart device such as a phone or commercially available fitness accessories. This communication is provided by the Wi-Fi peripheral inside ESP32 or the external ANT+ module D52 by GARMIN.

The power management block is shown in Figure 4 and it consists of four power lines. The external accumulator has a nominal voltage of $U = 18\,\mathrm{V}$ but the radar needs to be supplied with a voltage of $U = 12\,\mathrm{V}$. Therefore the RT2589 DC/DC step-down converter is used. For supplying LEDs the AP63205 DC/DC converter with fixed output voltage $U = 5\,\mathrm{V}$ was chosen. The same converter but with a different fixed output voltage of $U = 3.3\,\mathrm{V}$ is used to supply all chips on the PCB.

Besides the fact that the USB connector serves as a programming interface for the ESP32, it also serves as a second power source. The seamless transition between $5\,\mathrm{V}$ converted from the battery voltage and the voltage from the USB is provided by the TPS2121 power multiplexer. When both sources are available the USB source has priority, but when it's not present the module is powered by the battery.

The input power line and also the $12\,\mathrm{V}$ power line are monitored by the INA3221 chip. It senses the current by measuring voltage on shunt resistors and it also measures voltages on these lines. From these two values, the power consumption can be calculated. This chip communicates with ESP32 by the I^2C bus. The maximum power consumption is calculated to be $P = 13.6\,\mathrm{W}$. The whole schematic diagram can be found in the GitHub repository.

IV. Software

The software for this project includes firmware in ESP32 and ROS2 middleware running on the SBC. The whole ROS2 graph can be seen in Figure 5. The SBC runs two ROS2

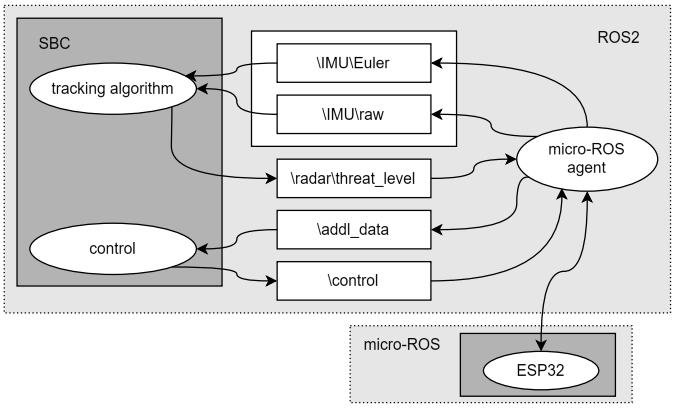

Fig. 5. RSS ROS2 graph (nodes are in ellipses, topics are in rectangles)

nodes. One of them, the *tracking algorithm*, subscribes to topics containing the Euler angles and raw data from the IMU. As an output, it publishes the threat level. The other node, the *control* node, subscribes to additional information, such as temperature, power consumption, and amount of ambient light, and publishes instructions for power management IC. All the topics use standard ROS2 messages. Development of this system was done on Microsoft Windows Subsystem for Linux (WSL) with running the Ubuntu distribution. All ROS2 principles were sourced from the ROS2 wiki [4].

The firmware for ESP32 was written in C language using the ESP-IDF framework. While ROS2 is a powerful tool for non-resource-constrained devices, for embedded devices it cannot be used. Fortunately, there exists a lightweight version of ROS2 for microcontrollers, called the *micro-ROS* [5]. It can be compiled for ESP32 but to communicate with the ROS2 graph it needs an *agent*, which runs also on SBC. The micro-ROS libraries were imported to the ESP-IDF environment as a component and were used to create the only node on ESP32. The whole firmware is supported by the FreeRTOS system.

The ESP32 takes the raw data from the IMU as linear acceleration and angular velocity and transforms them to Euler angles using the complementary filter. In further development, the Kalman filter will be used [6]. The node publishes both the raw data and the Euler angles. On the other hand, the subscribed threat level is used to light up the LEDs or beep the buzzer. The additional information on power consumption can be calculated from voltage and current values obtained from the power monitor IC. This information together with others previously mentioned is published to the *\addl_data* topic. The voltage converter for the radar can be disabled according to the message from the *\control* topic.

V. Hardware design

The PCB was designed in KiCad freeware software. Since the design includes 50 MHz clock lines and differential pairs, the impedance control and delay matching principles had to be followed when designing the PCB. The PCB has four layers with the following stack-up: high-speed traces, ground plane, power domains, and low-speed signals. The visuals of the PCB can be found in the GitHub repository.

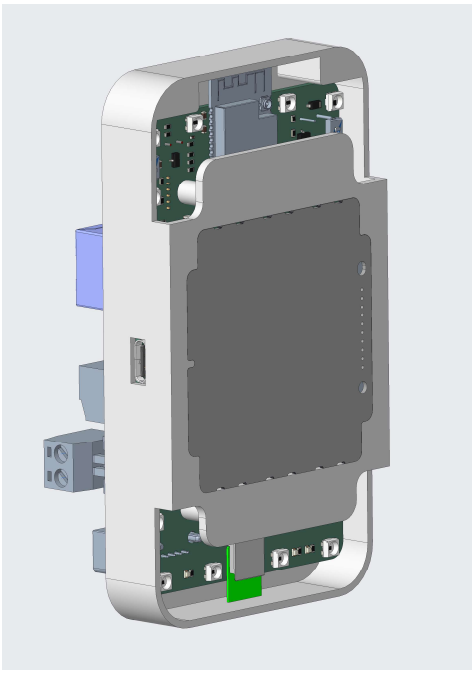

Fig. 6. 3D design of the enclosure with the module and radar board

The ALPS Generic Radar 5 is connected to the development module by spring-loaded pins and Flexible flat cable (FFC) used for future debugging. Both boards are enclosed in 3D printed covering mounted on the seat pole of a bicycle. The design of the complete device is shown in Figure 6.

VI. Conclusion

The development module was designed, manufactured, and assembled according to the needs of the company. The software concept was presented and is currently being programmed. Finally, the ALPS development team will use this module for future development of the ALPS Generic Radar 5, but mainly to prove the concept of enhancing the tracking algorithm by the external IMU data.

References

[1] P. Norman, "What are rearview radar bike lights and should you use one?", *Bike Radar*, 2023.
[2] N. H. Khan and A. Adnan, "Ego-motion estimation concepts, algorithms and challenges: an overview", *Multimedia Tools and Applications*, vol. 76, no. 15, pp. 16581-16603, 2017.
[3] L. Seongwook, S. Lee, S. Lim, and S. -C. Kim, "Machine Learning-Based Estimation for Tilted Mounting Angle of Automotive Radar Sensor", *EEE Sensors Journal*, vol. 20, no. 6, pp. 2928-2937, 2020.
[4] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild", *Science Robotics*, vol. 7, no. 66, May 2022.
[5] P. Nguyen, "MICRO-ROS FOR MOBILE ROBOTICS SYSTEMS", Master thesis, Västerås, 2022.
[6] H. Ferdinando, H. Khoswanto, and D. Purwanto, "Embedded Kalman Filter for Inertial Measurement Unit (IMU) on the ATMega8535", in *2012 International Symposium on Innovations in Intelligent Systems and Applications*, 2012, pp. 1-5.