July 10, 2022
Dr. Craig Scratchley
School of Engineering Science
Simon Fraser University
British Columbia, V5A 1S6

Re: ENSC405W/440 Design Specification for Echo's Proximity Entrance System™

Dear Dr. Craig Scratchley,

In the following document you will find the design specifications used to implement Echo's Proximity Entrance System™ - a secure remote keyless system used by motorcycle riders to automatically start their vehicles. When set up, the Proximity Entrance System™ will analyze an approaching rider's intent to predict whether they will want to start the bike, issuing an early-warning message to the vehicle to prepare its relevant systems.

This design specification details the hardware and software implementations that will be used to implement the relevant functionality of the Proximity Entrance System™. Said functionality corresponds to the requirements outlined in the earlier requirements document.

Team Echo is composed of five computer engineering students - Shawn Baltar, Olivia Kuninaka, Spencer Pauls, Miller Solis, and Spencer Karjala - that have been collaborating to realize the Proximity Entrance System™ for the past couple months.

Thank you for your time and your effort spent reviewing this design specification. Questions or comments can be directed towards Spencer Karjala at skarjala@sfu.ca.

Sincerely,

[illegible]

Spencer Karjala
Chief Executive Officer
Echo

# PES

# PROXIMITY ENTRANCE SYSTEM™

## Design Specification

**Partners**:

Shawn Baltar
Olivia Kuninaka
Spencer Pauls
Miller Solis

**Submitted to**:

Dr. Craig Scratchley, ENSC405W
Dr. Andrew Rawicz, ENSC440
School of Engineering Science
Simon Fraser University

**Issue date**:

July 14, 2022

## Abstract

Modern remote keyless entry systems do not account for the intent of their user when automatically unlocking or locking the vehicle as the user approaches, leading to false starts. Furthermore, an unlock mechanism works until the user sits down, after which they may need to wait a period of time before they can start driving their vehicle. For premium electric motorcycles, users expect a premium experience - they want to sit down and immediately start moving. To resolve these issues, the Proximity Entrance System™ (PES) analyzes an approaching user's intent to determine whether they are looking to start or lock their vehicle, then issues correct messages in advance to the motorcycle to prepare its systems for an instant start. This document fully specifies the implementation of the PES™, its communication protocols, hardware, software, security practices, and how the system facilitates a premium user experience for the rider.

## Table of Contents

## Changelog

| Version | Description of Change | Date Modified | Modified By |
|---|---|---|---|
| 1.0 | Initial Document | July 14/2022 | Team Echo |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## List of Figures and Tables

## Glossary

**AES:** Advanced Encryption Standard; A block cipher encryption method

**Asymmetric encryption:** A cryptographic system that uses two separated (but algorithmically related) public and private key to encrypt and decrypt data

**CAN**: Controller Area Network; a physical-layer communication standard used for priority communications in automotive systems

**CLI:** Command-line Interface; a text-based interface used to run programs, manage computer files and interact with the computer

**ECU:** Electronic Control Unit; an embedded system in an automotive electrical system that is used to control a single component of the vehicle

**EUID**: Extended Unique Identifier; a unique 64-bit identifier associated with MAC IEEE 802.15.4

**GCM:** Galois/Counter Mode; an encryption algorithm associated with the AES encryption method

**GPIO**: General-Purpose Input/Output; a standard interface used to connect microcontrollers to any peripheral devices

**IC**: Integrated Circuit; an electronic circuit implemented on a single silicon chip

**IT:** Information Technology; A support team at a company to aide in the user of computers, storage, networking, and all other forms of electronic data

**MAC**: Medium Access Control; the transmission layer that interfaces with physical transmission media

**MCU**: Microcontroller Unit; a miniature computer implemented on a single semiconductor chip

**OEM:** Original Equipment Manufacturer; a company that supplies manufactured parts that are used in another company's larger product system

**PAN**: Personal Area Network; a local computer network used to interconnect a user's devices

**PCB:** Printed Circuit Board; a laminated structure used for condensed wiring in electrical circuits

**PDM:** Proximity Detection Module; the bike-mounted embedded system component of the Proximity Entrance System™

**PII:** Personally Identifiable Information; any data that could potentially identify a specific individual

**PES**: Proximity Entrance System™; an automotive system that predicts user intent to reduce the time waiting for an electric motorcycle to start

**PLA**: Polylactic Acid; material used in 3D printing

**Replay Attack**: A type of network attack where an attacker detects a data transmission and fraudulently has it delayed or repeated

**RID:** Remote Identifier; the user-held wearable component of the Proximity Entrance System™

**RSSI**: Received Signal Strength Indicator; a part of Bluetooth that may be used to implement RTLS using signal strength measurements

**RTLS**: Real-Time Locating System; a system used to track the position of people or objects in real time

**SPI**: Serial Peripheral Interface; a communication bus that facilitates data transfer often between microcontrollers and their attached peripherals

**UWB**: Ultra-Wideband; a wireless technology used to communication position data to a high accuracy in a local area

# 1. Introduction

## 1.1 Background

Echo is developing the Proximity Entrance System™ (PES) to provide the ultimate user-friendly experience for motorcyclists who want to sit down and start riding. This means as little interaction as possible - the PES™ aims to anticipate the rider's intent and to be ready to go at any given moment. By utilizing a bike-mounted embedded device, paired with a small and user-friendly portable component, the PES™ will use proximity and intent detection to notify the bike of an impending boot.

As the PES™ is a single module, Echo aims to license this device to motorcycle manufacturers looking to maximize their user experience by minimizing wait times. By viewing the bike as a black box, the PES™ will come with a simple interface to be integrated into an electric motorcycle. By focusing on the rapidly growing market of electric motorcycle manufacturing [1], the user experience afforded by the PES™ is positioned to become the new standard for vehicle entry and exit.

## 1.2 Overview

The PES™ is built to challenge the problem of waiting for vehicle boot times by reducing them to zero while minimizing the number of accidental starts. The system is split into two components: the Proximity Detection Module (PDM), which is mounted on the bike, and a Remote Identifier (RID), which is held by the user.

As a user approaches their bike while holding their RID, the PDM and RID communicate to determine the RID location. When the RID approaches the bike with intent to ride, the PDM sends a "wake-up" signal to the system - not necessarily to perform a full boot, but to give the system an advanced warning that a user is approaching. This provides the system with a chance to run through any lengthy boot-up processes so the user can get on and go.

In addition to a “start” and “lock” signals triggered based on user input combined with positioning and trajectory information, the PES™ allows for a seamless and secure user interaction, even when walking away from the motorcycle.

A general view of this architecture is shown in Figure 1.2.1.

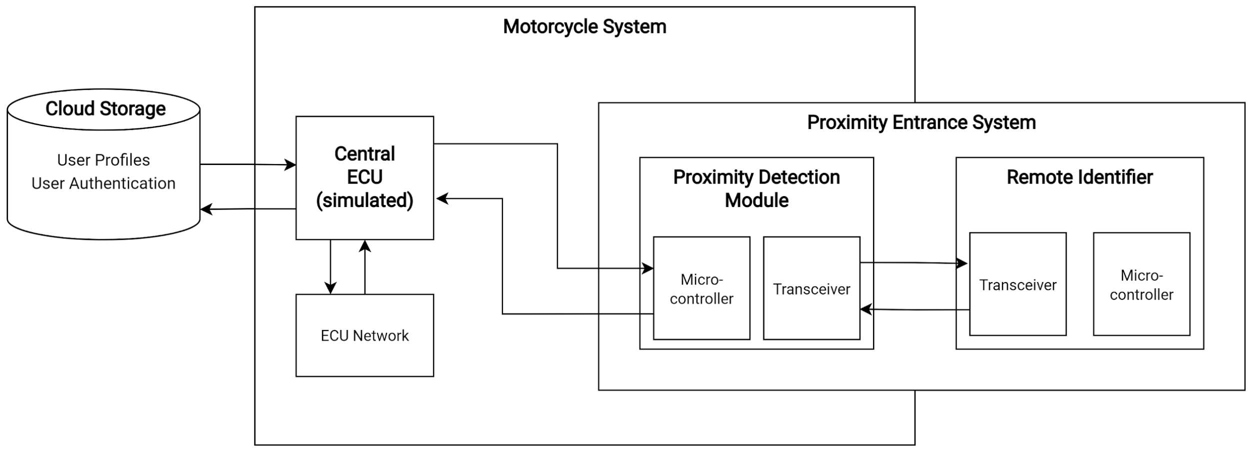


Fig. 1.2.1: PES™ Architecture

## 1.3 Design Challenges

The main challenge in designing the PES™ will be implementing intent detection through wireless communications between the PDM and potentially multiple RIDs. The PDM will need to have two antennas to perform triangulation with each RID, and these antennas will need to be spaced as far apart as possible to maximize accuracy. This also means that a wireless protocol will have to be created that can securely identify each RID. Because there are multiple antennas in close proximity, extra effort will have to go into tuning the algorithm to deal with interference effects and to maximize intent detection reliability.

Another challenge will come in the physical and embedded design of the RID. It needs to consume as little power as possible to maximize its battery life while also enabling communications with its registered PDM. Furthermore, it must be physically small enough to fit inside of a user's pocket or to be secured as a wearable device while including all of the necessary hardware.

One final challenge will be to perform all communications and actions securely. As a keyless entry system, it is extremely important that potential attackers cannot gain entry to the vehicle by listening to or manipulating communications between the PDM and RID, or between the PDM and the bike itself. As a result, encryption schemes will need to be designed for all communications to ensure that only a registered user may start the vehicle.

## 1.4 Scope

This document aims to completely specify the design specifications of the PES™ through its alpha, beta, and planned release product development stages. Individual design specifications along with appropriate justifications will be provided for the hardware, software, communications, security, and physical components of the system. Alternative solutions to the design options chosen will be provided in Appendix B.

## 1.5 Design Classification Format

All design specifications in this document will be specified as follows:

`[D-W.X.Y.Z.V]`

This format can be interpreted as follows:

- `W` is the design class;
- `X` is the design subclass;
- `Y` is the design device, where '0' is for the entire product, '1' is for the RID, '2' is for the PDM, and 3 is for the simulated ECU;
- `Z` is the design number; and
- `V` is the product development version, where 'a' is for alpha phase, 'b' is for beta phase, and 'c' is for the planned release phase.

## 2. High Level Overview

### 2.1 Remote Identifier (RID)

The Remote Identifier (RID) is the user-held component of the Proximity Entrance System™ (PES) used for two purposes: uniquely identifying each rider, and communicating position information with the Proximity Detection Module (PDM). Each RID has an ultra-wideband (UWB) transceiver that it uses to communicate with the PDM installed on the motorcycle and a low-power microcontroller used to coordinate secure, encrypted communication with the PDM. It also includes a backup activation button in case of software intent detection failure.

In the planned product release stage, the RID will have a low-profile casing used to mount it to various surfaces. For example, a user could potentially mount their RID to their helmet or leave it in their purse or wallet to reduce the number of items they need to carry.

### 2.2 Proximity Detection Module (PDM)

The Proximity Detection Module (PDM) of the PES™ is the main embedded system installed into the motorcycle's ECU network. It consists of a microcontroller, a CAN bus, and a pair of UWB transceivers. The microcontroller handles all proximity detection logic, encrypted communication, security logic, and database communication; the CAN bus is used to interface with the motorcycle's central control unit; and the UWB transceivers are used to triangulate the position of any number of registered RIDs in the local area of the PDM.

In the planned product release stage, the PDM will be contained in a low-profile casing to simplify installation into a motorcycle by its manufacturer. The antennas may be contained within this housing or wired to be installed elsewhere on the motorcycle to maximize connectivity and data accuracy. Specifications for the casing will be provided to any manufacturer licensing the product in the likely event that they wish to customize its physical design.

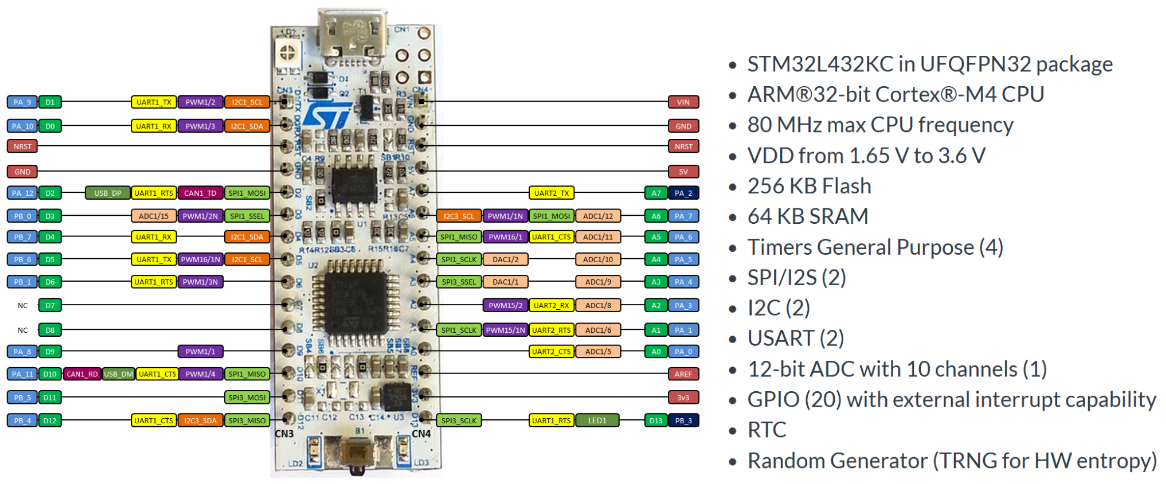
## 3. Hardware

The PES™ is primarily an embedded device. The majority of its important functionality comes from its ability to communicate between the different components that make up its system, and from the antennas used for position sensing on those components. Because of the form factor, these hardware components must be chosen to minimize power consumption while maximizing performance.

### 3.1 Remote Identifier (RID)

The RID must be capable of secure, encrypted communication through its ultra-wideband (USB) transceiver. To coordinate this, a low-power microcontroller is included in its design.

The STM32 NUCLEO-L432KC is a 32-bit ARM-based microcontroller development board that prioritizes ultra low-powered performance while maintaining a broad range of functionality for prototyping. This unit has been selected to implement the RID due to this balance of performance and functionality, combined with its low cost. This development board supports SPI communication through its GPIO pins, so it can be connected to the RID's UWB transceiver to implement wireless communication with the PDM. Figure 3.1.1 shows the pinout and product specifications of the NUCLEO-L432KC, and Table 3.1.1 outlines the exact specifications required for the RID microcontroller.



- STM32L432KC in UFQFPN32 package
- ARM®32-bit Cortex®-M4 CPU
- 80 MHz max CPU frequency
- VDD from 1.65 V to 3.6 V
- 256 KB Flash
- 64 KB SRAM
- Timers General Purpose (4)
- SPI/I2S (2)
- I2C (2)
- USART (2)
- 12-bit ADC with 10 channels (1)
- GPIO (20) with external interrupt capability
- RTC
- Random Generator (TRNG for HW entropy)

Fig. 3.1.1: STM32 NUCLEO-L432KC Pinout [2]

In the planned product release, the RID will be a wireless device, which means it must be battery-powered. Due to the difficulty of developing a power supply for the STM32 NUCLEO-L432KC itself, and because said power supply would be extremely different from the battery power solution in the final product, the alpha prototype of the RID will be powered using the built-in USB power for the NUCLEO-L432KC.

| Design ID | Specification Description | Referenced Requirement |
|---|---|---|
| D-3.1.1.1.a | The RID microcontroller will be able to communicate with a single UWB transceiver. | R-4.1.1.1a |
| D-3.1.1.2.b | The RID microcontroller will be able to communicate with LEDs to signal a successful motorcycle start. | R-4.1.1.5.b |
| D-3.1.1.3.b | The RID microcontroller will be able to communicate with a small speaker to signal a successful motorcycle start. | R-4.1.1.5.b |
| D-3.1.1.4.a | The RID microcontroller will have a button to notify the PDM to lock the motorcycle. | Not done |
| D-3.1.1.5.a | The RID microcontroller will have non-volatile storage to hold one preprogrammed encryption key. | R-4.1.1.2.a, R-4.1.1.3.a |

Table 3.1.1: RID Microcontroller Design Specifications

In the final release, it is expected that the RID will be working on a custom PCB. In this scenario, it must have a competitive [3] battery life of at least 12 months using batteries that are common and easy to replace. For a premium user experience, the RID should also signal when its batteries should be changed using its onboard LEDs. Table 3.1.2 has the design specifications for the RID's battery power.

| Design ID | Specification Description | Referenced Requirement |
|---|---|---|
| D-3.1.1.6.c | The RID hardware will use standard 3V button-cell batteries. | R-3.4.1.1.a |
| D-3.1.1.7.c | The RID hardware will run for at least 12 months on its included batteries. | R-3.4.1.2.b, R-7.4.1.1.b |

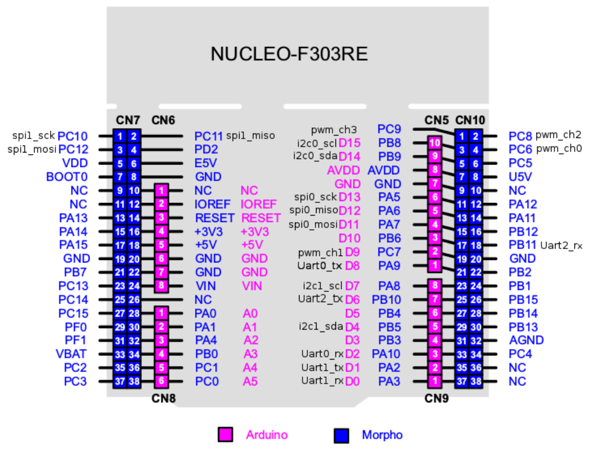
Table 3.1.2: RID Battery Power Design Specifications

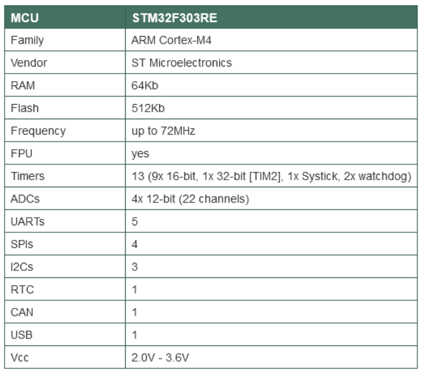
## 3.2 Proximity Detection Module (PDM)

Like the RID, the PDM must be able to perform secure, encrypted communication - however, it must do this over both UWB and CAN. In addition, the PDM must also coordinate intent detection logic, security logic, and database communication. For this reason, a more powerful microcontroller has been selected for the PDM.

The STM32 NUCLEO-F303RE is a 32-bit ARM-based microcontroller development board that provides more performance at the cost of a higher power consumption. This unit was selected to implement the

PDM as it is powerful enough to implement all of the central control logic that is needed to drive the PES™ system, and because it has a built-in CAN controller and SPI support. This means the board can facilitate connection to a CAN bus and also connect to UWB transceivers to communicate with nearby RID units. Figure 3.2.1 shows the pinout and product specifications of the NUCLEO-F303RE, and Table 3.2.1 contains the design specifications required for the PDM microcontroller.





| MCU | STM32F303RE |
|---|---|
| Family | ARM Cortex-M4 |
| Vendor | ST Microelectronics |
| RAM | 64Kb |
| Flash | 512Kb |
| Frequency | up to 72MHz |
| FPU | yes |
| Timers | 13 (9x 16-bit, 1x 32-bit [TIM2], 1x Systick, 2x watchdog) |
| ADCs | 4x 12-bit (22 channels) |
| UARTs | 5 |
| SPIs | 4 |
| I2Cs | 3 |
| RTC | 1 |
| CAN | 1 |
| USB | 1 |
| Vcc | 2.0V - 3.6V |

Fig. 3.2.1: STM32 NUCLEO-F303RE Product Specifications [4]

| Design ID | Specification Description | Referenced Requirement |
|---|---|---|
| D-3.2.2.1.a | The PDM microcontroller will be able to communicate with two UWB transceivers. | R-4.2.2.1.a |
| D-3.2.2.2.a | The PDM microcontroller will be able to send and receive communications over a CAN bus. | R-4.2.2.3.a |
| D-3.2.2.3.a | The PDM microcontroller will have non-volatile storage to hold two preprogrammed encryption keys. | R-4.2.2.2.a |

Table 3.2.1: PDM Microcontroller Design Specifications

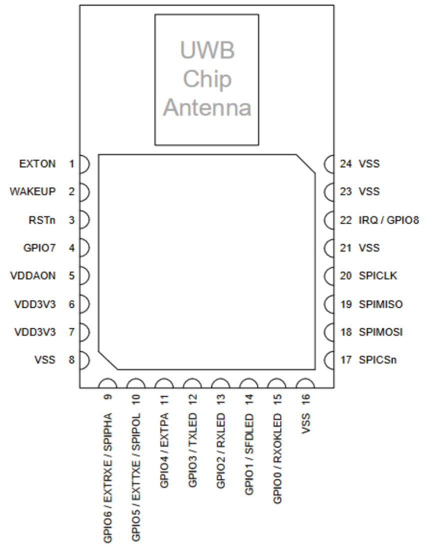
Similar to the microcontroller used for the prototype RID, designing and building a power supply for the NUCLEO-F303RE is out of scope. The final device will be powered by a motorcycle; designing and porting a custom USB power system to an analog power supply would be wasted effort. As a result, the prototype of the PDM will be built using the microcontroller development board's onboard USB power. Table 3.2.2 specifies how the PDM will be powered in the final product development phase.

| Design ID | Specification Description | Referenced Requirement |
|---|---|---|
| D-3.2.2.4.a | The PDM will be powered through a standard 12V input. | R-3.4.2.1.a |

Table 3.2.2: PDM Power Design Specifications

## 3.3 Ultra-Wideband (UWB)

In the PES™, both the PDM and RID require access to at least one ultra-wideband (UWB) transceiver to implement high-precision device location. In the case of the PDM, a connection to two UWB transceivers is needed to facilitate triangulation. For this purpose, the DecaWave DWM1000 RF transceiver was chosen for the final product phase. The DWM1000 can work on an SPI bus, meaning that multiple transceivers can work as slave devices. It is also a low-power chip, meaning it will work well towards the RID's battery life requirements, and it includes hardware MAC addresses for each module. Figure 3.3.2 shows the pinout and specifications for the DWM1000.



- IEEE 802.15.4-2011 UWB compliant
- Supports 4 RF bands from 3.5GHz to 6.5GHz
- Programmable transmitter output power
- Fully coherent receiver for maximum range and accuracy
- Designed to comply with FCC & ETSI UWB spectral masks
- Supply voltage 2.8V to 3.6V
- Low power consumption
- Data rates of 110kbps, 850kbps, 6.8Mbps
- Maximum packet length of 1023 bytes for high data throughput applications
- Integrated MAC support features
- Supports 2-way ranging and TDOA
- SPI interface to host processor
- 23mm x 13mm x 2.9mm 24-pin side castellation package

Fig. 3.3.1: DecaWave DWM1000 UWB Transceiver Specifications [5]

For prototyping, the DWS1000 will be used, which is the development board provided by DecaWave for the DWM1000. This board simplifies the process of prototyping with the DWM1000, since developers can rearrange connections without needing to re-solder connections and potentially break their development hardware. An example board with its pinout is shown in Figure 3.3.2, and an example of wiring the DWM1000 with the RID's NUCLEO-L432KC development board is shown in Figure 3.3.3. The design specifications for the PES™'s UWB transceiver are listed in Table 3.3.1.

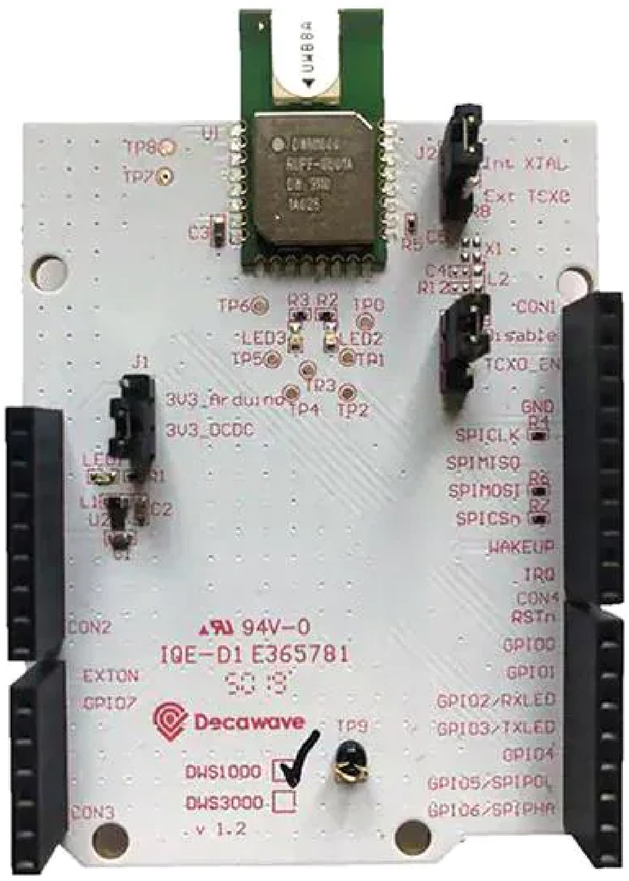


Fig. 3.3.2: DecaWave DWS1000 UWB Development Board Pinout [6]

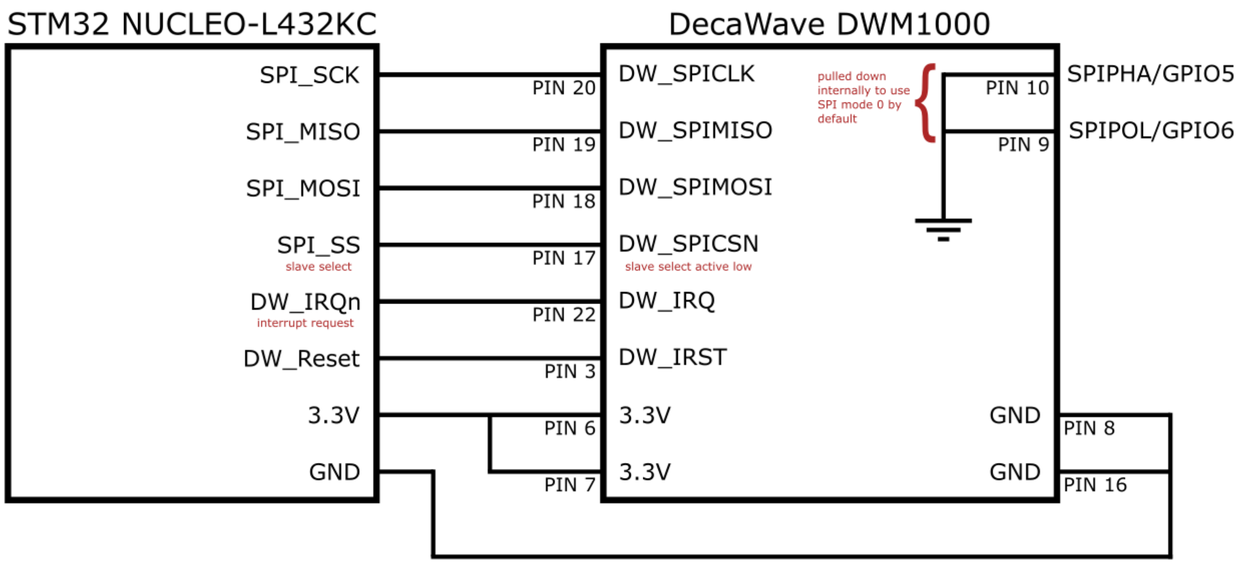


Fig. 3.3.3: Example Wiring for DWM1000 with RID NUCLEO-L432KC Prototype

| Design ID | Specification Description | Referenced Requirement |
|---|---|---|
| D-3.3.0.1.a | The UWB module will be able to have bidirectional communication with another identical UWB module. | R-4.1.1.1.a, R-4.2.2.1.a |
| D-3.3.0.2.a | The UWB module will be able to communicate time-of-flight information with another identical UWB module. | R-4.1.1.1.a, R-4.2.2.1.a |
| D-3.3.0.3.a | The UWB module will contain a unique identifier. | R-5.3.1.1.a, R-5.3.1.3.a, R-5.3.2.4.a |

Table 3.3.1: PES™ UWB Module Design Specifications

## 3.4 Controller Access Network (CAN) Bus

In automotive products, a Controller Area Network (CAN) bus is the standard communication bus for onboard ECUs, which are used to manage each individual system on the vehicle. A node on a CAN bus needs two components: a CAN controller, and a CAN transceiver. The CAN bus itself is composed of two wires labeled CANL (CAN low) and CANH (CAN high).

CAN uses a differential signal to communicate information over the bus. Specifically, a '1' is encoded by a "dominant" state where CANH > CANL, and a '0' is encoded by a "recessive" state where CANL < CANH [7]. To achieve the required voltage differences, a 120Ω resistor is placed between CANL and CANH at terminating nodes. A simple example is shown in Figure 3.4.1.

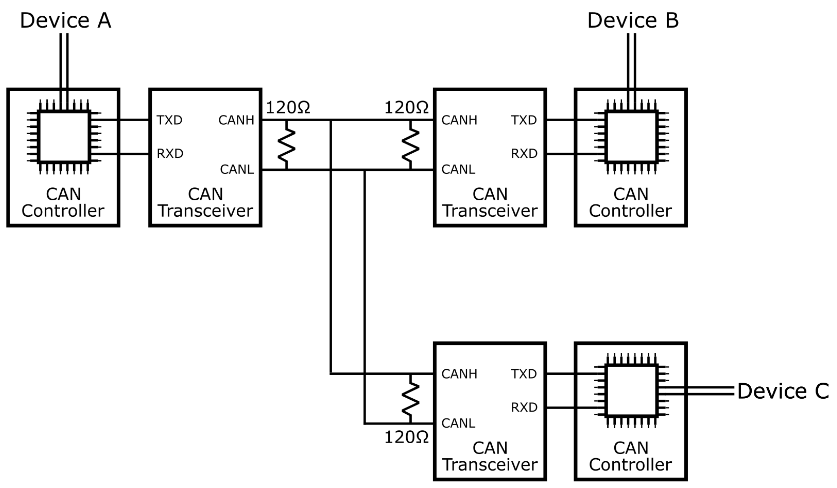


Fig. 3.4.1: Example CAN Bus Topology

## 3.4.1 Simulated ECU

On the simulated ECU side, a CAN-USB interface is needed to translate the real physical-layer CAN messages to serial data so that the simulation software can interpret it. The Seeed Studio USB to CAN Analyzer was chosen for this due to its low price and having the benefit of not needing any additional hardware, since both the CAN controller and the CAN transceiver are included. It also comes with simple drivers to have the bus appear as a serial device in both Linux and Windows machines. The module is pictured in Figure 3.4.2 with a labeled pinout, and the required specifications for the CAN connection used by the simulated ECU are listed in Table 3.4.1.

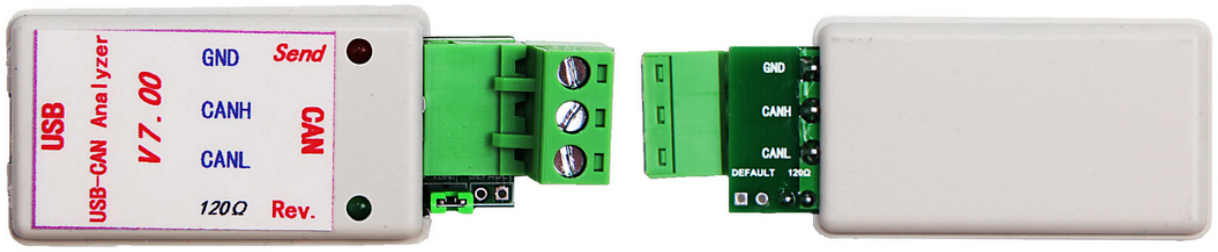


Fig. 3.4.2: Seeed Studio USB to CAN Analyzer [8]

| **Design ID** | **Specification Description** | **Referenced Requirement** |
|---|---|---|
| D-3.4.3.1.a | The simulated ECU CAN interface will provide bidirectional software communications with the CAN bus. | R-4.2.2.3.a, R-4.2.2.5.a |
| D-3.4.3.2.a | The simulated ECU CAN interface will work with Linux machines. | R-4.2.2.3.a, R-4.2.2.5.a |

Table 3.4.1: Simulated ECU CAN Interface Design Specifications

## 3.4.2 Proximity Detection Module

The PDM will be connected to a simulated ECU in software by a CAN bus. This way, connection to a manufacturer's motorcycle network will be as simple as possible. As mentioned in Section 3.2, the microcontroller used for the PDM contains a CAN controller, so a CAN transceiver is needed to facilitate connection. For development, the Waveshare SN65HVD230 development board was chosen for its low cost and ease of use - an example with pinout is shown in Figure 3.4.3.

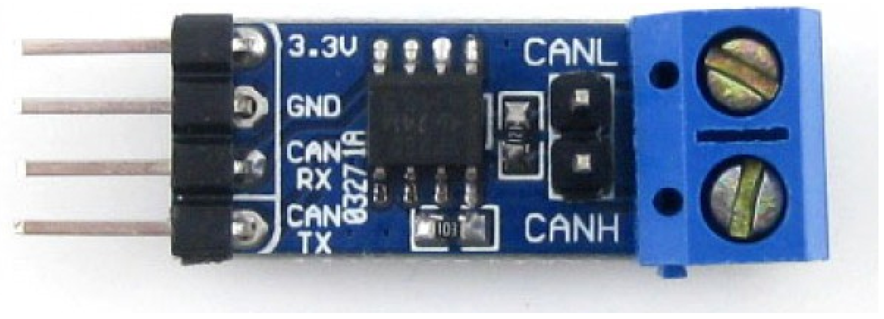
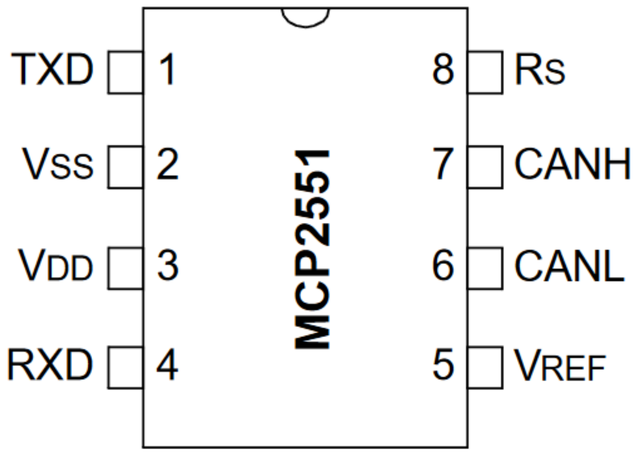


Fig. 3.4.3: Waveshare SN65HVD230 Development Board Pinout [9]

For the planned product release stage, the Microchip MCP2551 has been selected as the CAN transceiver for the PDM as it is a standard part for this role, is inexpensive, and has a large amount of support. Its pinout and specifications are pictured in Figure 3.4.4.



| | |
|---|---|
| CAN FD | No |
| TX Buffers | 0 |
| Rx Buffers | 0 |
| Masks | 0 |
| Filters | 0 |
| Operating Temperature Max. (C) | +125 |
| Operating Temperature Min. (C) | -40 |
| Operating Voltage Max. (V) | 5.5 |
| Operating Voltage Min. (V) | 4.5 |

Fig. 3.4.4: Microchip MCP2551 Product Specifications [10]

A circuit for the CAN bus with the PDM and simulated ECU is shown in Figure 3.4.5 using the MCP2551. Note that $V_{REF}$ is left open and $R_S$ is tied to ground across a load resistor that controls rise/fall times. Table 3.4.2 shows the specifications needed by the PDM's CAN hardware to implement required functionality.

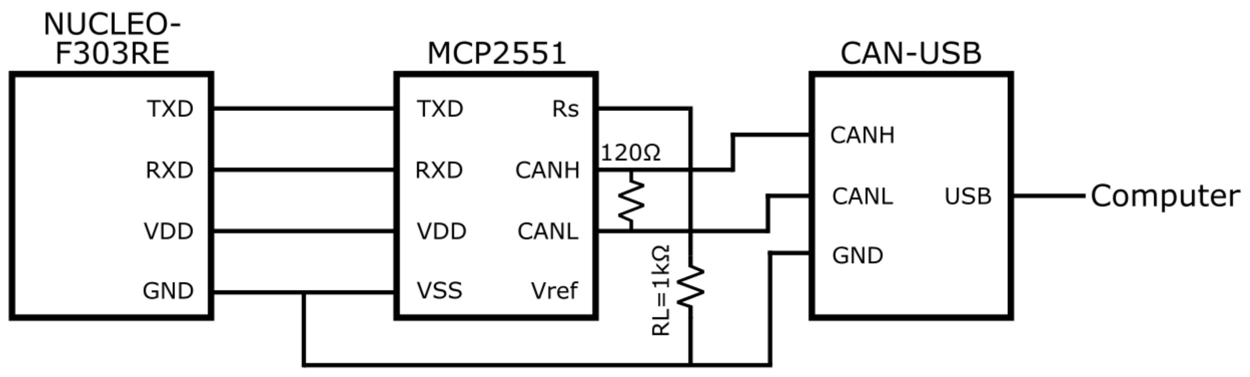


Fig. 3.4.5: PDM-ECU CAN Bus Circuit

| **Design ID** | **Specification Description** | **Referenced Requirement** |
|---|---|---|
| D-3.4.2.1.a | The PDM CAN transceiver will implement CAN message transfer and receival with the simulated ECU over the CAN bus. | R-4.2.2.3.a, R-4.2.2.5.a |
| D-3.4.2.2.a | The PDM CAN transceiver will be powered by a standard 3.3V or 5V microcontroller source voltage. | R-4.2.2.3.a, R-4.2.2.5.a |
| D-3.4.2.3.c | The PDM CAN controller and transceiver pair will fully implement CAN communications with a motorcycle ECU over the CAN bus. | R-4.2.2.4.c |

Table 3.4.2: PDM CAN Hardware Design Specifications

# 4. Communication

## 4.1 Ultra-Wideband (UWB) Communication

Because UWB is simply a radio communication technology, a communication protocol must be implemented to facilitate data transfer. A full overview of the UWB communication protocol used between the PDM and RID is shown in Figure 4.1.9. This protocol and the following frame standards are based off of the APS013 application note provided by DecaWave [11]. As indicated in the application note, these frame encodings follow the IEEE 802.15.4-2011 standard [12].

### 4.1.1 Preamble

Each frame below contains a *Frame Control* (FC) field as defined by the IEEE 802.15.4-2011 standard [12]. This field is defined as shown in Figure 4.1.1.

- *Frame type* is used to indicate what type of frame is being sent
- *Security enabled* (SEC) indicates whether MAC sublayer security is used, so this is '0'
- *Frame Pending* (PEND) indicates whether more data will be sent
- *Acknowledgement Request* (AR) indicates whether an acknowledgement is necessary
- *PAN ID Compression* (PIC) controls whether a source and destination Personal Area Network (PAN) ID are needed; set to '1' to require only a destination PAN ID
- *Destination Address Mode* (DestAdrMd) specifies if a 16-bit or 64-bit destination address is used
- *Version* indicates the frame version, which is "00" for this case
- *Source Address Mode* (SrcAdrMd) specifies whether a 16-bit or 64-bit source address is used

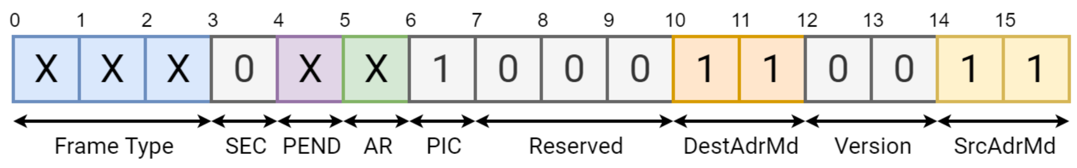


Fig. 4.1.1: UWB Frame Control Field Format

### 4.1.2 Listening Phase

The initial *listening phase* occurs while the RID is out of range of the PDM. During this phase, the RID sends *blink messages* that follow the format in Figure 4.1.2. During this time, when a blink message is not received, the RID enters a sleeping state for a predefined time period, then sends another blink message.

- *Frame control* is 0x0200, since the frame type is "000" for a beacon message, PEND is '0' for no pending data, AR is '0' for no ACK required, destination address is "11" for a full 64-bit EUID, and source address is "00" as it is not needed
- *Sequence number* corresponds to the increment counter for beacon frames
- *PDM MAC EUID* is the identifier used by the receiving PDM to filter out irrelevant RID frames
- *FCS* is the *Frame Check Sequence*, used similarly to a CRC to verify received frame integrity, which is set and checked automatically by the DWM1000 UWB transceivers

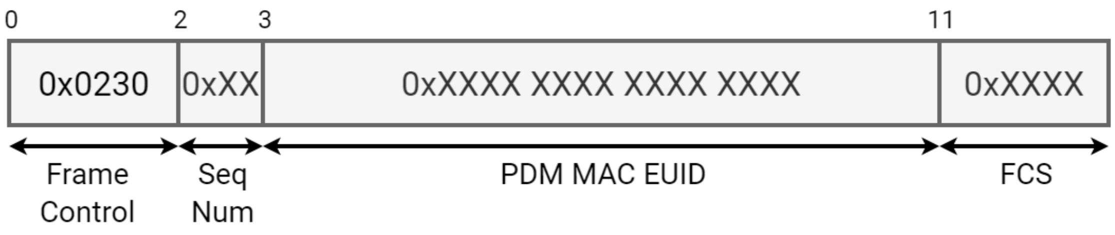


Fig. 4.1.2: UWB Blink Message Format

Once a PDM has identified an RID with which it was previously paired, it replies with a *ranging init* message. The format of this frame is shown in Figure 4.1.3.

- *Frame control* is 0x2232, since the frame type is "001" for a data message, PEND is '0' for no pending data, AR is '0' for no ACK required, destination address is "11" for a full 64-bit EUID, and source address is "10" for a short 16-bit address
- *Sequence number* corresponds to the increment counter for data frames
- *PAN ID* is the proposed identifier for the PAN being set up
- *RID MAC EUID* is the identifier of the registered RID
- *PDM short address* is the new short address the PDM will use for ranging
- *Function code* is 0x20 to indicate a data section format  for the init range frame
- *RID short address* is the proposed short address for the RID to use for ranging
- *Response delay* is the time to wait between successive ranging requests
- *FCS* is the CRC check, identical to the previous frame type

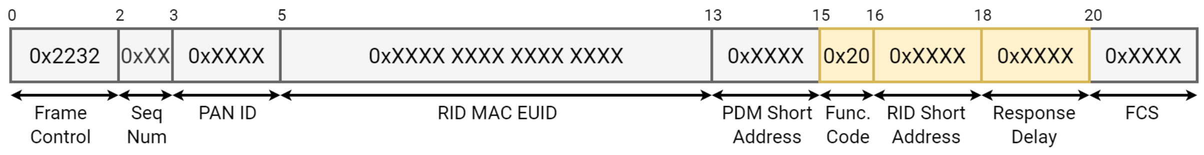


Fig. 4.1.3: UWB Ranging Init Message Format

### 4.1.3 Positioning Phase

Once the PDM has sent its ranging init message, the *positioning phase* begins. During this phase, there are three frame types: the poll frame, response frame, and final frame. Each of these frames has an identical message format with a different data section. The general frame format is shown in Figure 4.1.4, where explanations are left out for fields whose purposes are unchanged from the previous frame.

- *Frame control* is identical to the ranging init, except the destination address is now "10" to use a short 16-bit address
- *PDM short address* is the first address generated by the PDM at the end of the listening phase and was signaled to the RID during the ranging init message
- *RID short address* is the second address generated by the PDM that was signaled to the RID in the data section of the ranging init message

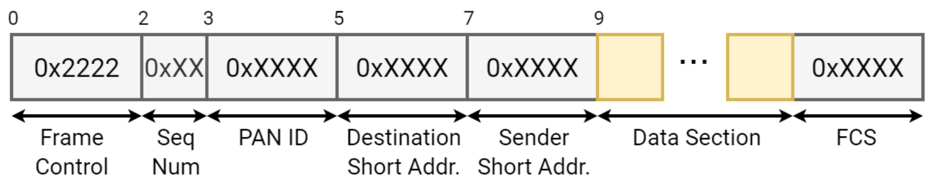


Fig. 4.1.4: General Positioning Message Format

The data sections for the three frames are pictured in Figure 4.1.5.

- *Poll* is indicated by a 0x61 function code, and is used by the RID to determine its Poll$_{TX}$ timestamp and by the PDM to determine its Poll$_{RX}$ timestamp
- *Resp* is indicated by a 0x50 function code, and is used by the PDM to determine its Response$_{TX}$ timestamp and by the RID to determine its Resp$_{RX}$ timestamp
- *Final* is used to transmit the timestamp results from the RID to the PDM, and also by the PDM to determine the last Final$_{RX}$ timestamp needed to determine the two-way ranging time of flight

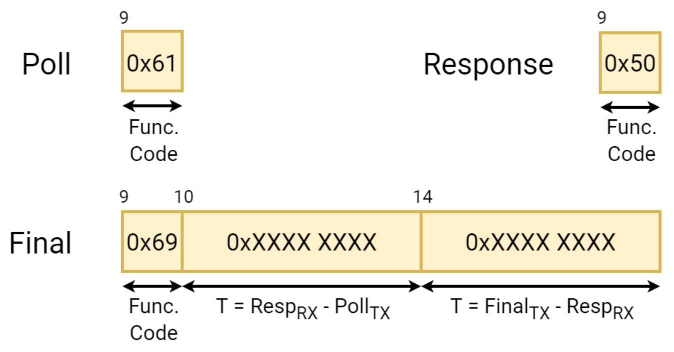


Fig. 4.1.5: Poll, Response, and Final Positioning Message Data Formats

## 4.1.4 Authentication Phase

Once the PDM has determined that the user intends to start the bike, the *authentication phase* begins. In this phase, there are three message types: an auth request message, an auth reply message, and an auth ACK message. The initial *authentication request* format is shown in Figure 4.1.6. The fields in this frame are identical in function to previous frames, with a new 0x80 function code to uniquely identify it.

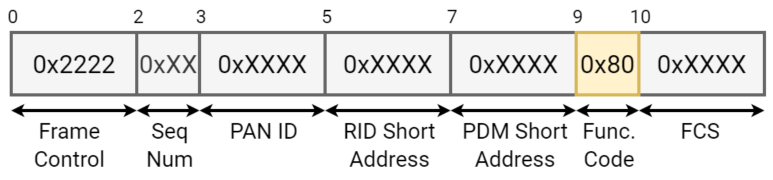


Fig. 4.1.6: Authentication Request Message Format

Once the RID has received an authentication request, it stops sending poll messages and enters the authentication phase. To reply, it takes its rolling code used to combat replay attacks (Section 6.1) and encrypts it with its symmetric AES-128 key, which is used to sign the *authentication reply* message shown in Figure 4.1.7.

- Note: the *ACK request* bit in the *frame control* field is not set although an ACK is required. This is because the UWB transceiver will send the ACK automatically, whereas the next ACK must be sent manually by the PDM once it has issued the "wake-up" action to the central ECU.

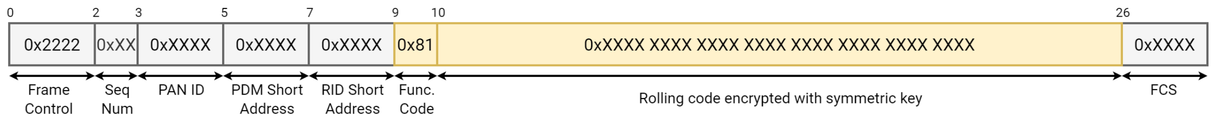


Fig. 4.1.7: Authentication Reply Message Format

The final *authentication ACK* message is sent by the PDM once it has issued the "wake-up" signal to the central motorcycle ECU. The PDM encodes the action performed (eg. whether the command succeeded) with its symmetric key and uses it to sign the message shown in Figure 4.1.8. This frame format is identical to the authentication request except for the swapped addresses and different data content.

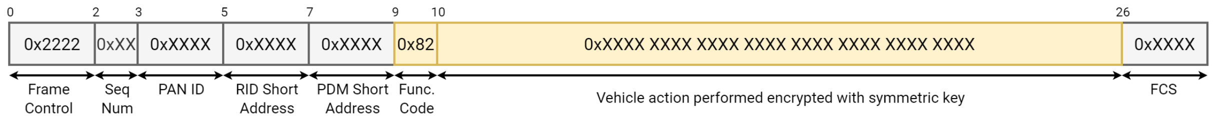


Fig. 4.1.8: Authentication ACK Message Format

## 4.1.5 Summary

The frames and the data being sent are summarized in Figure 4.1.9. The positioning phase cycle of poll, response, and final messages continues indefinitely until one of two things happen: either the PDM successfully reads the intent of the user to which the RID belongs, or the RID goes out of range and the positioning phase times out. This behavior is specified in Sections 5.1 and 5.2.

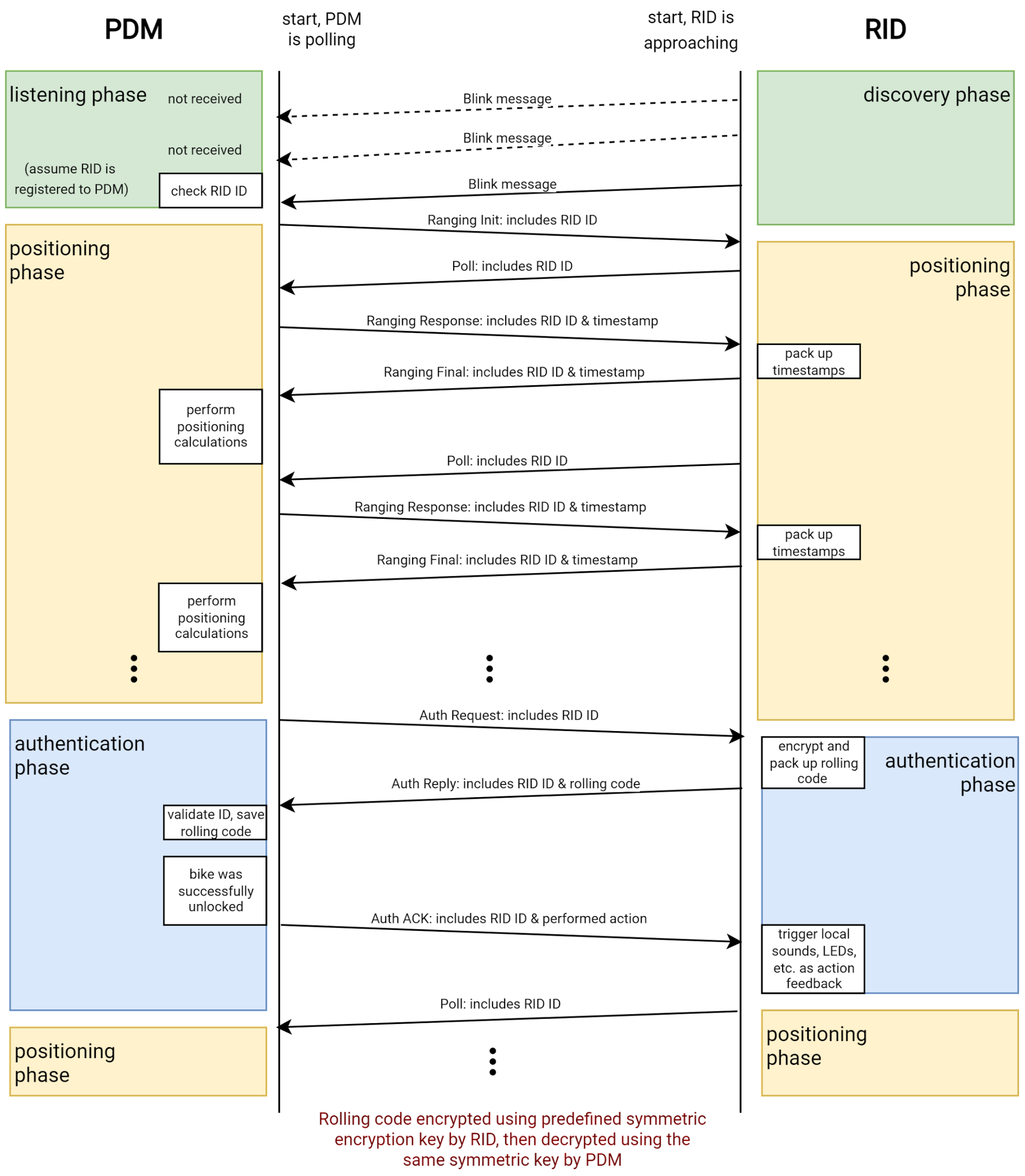

PDM
start, PDM is polling
start, RID is approaching
RID
listening phase
not received
not received
(assume RID is registered to PDM)
check RID ID
discovery phase
Blink message
Blink message
Blink message
Ranging Init: includes RID ID
Poll: includes RID ID
Ranging Response: includes RID ID & timestamp
Ranging Final: includes RID ID & timestamp
positioning phase
pack up timestamps
perform positioning calculations
Poll: includes RID ID
Ranging Response: includes RID ID & timestamp
Ranging Final: includes RID ID & timestamp
pack up timestamps
perform positioning calculations
Auth Request: includes RID ID
authentication phase
encrypt and pack up rolling code
Auth Reply: includes RID ID & rolling code
validate ID, save rolling code
bike was successfully unlocked
Auth ACK: includes RID ID & performed action
trigger local sounds, LEDs, etc. as action feedback
Poll: includes RID ID
positioning phase
Rolling code encrypted using predefined symmetric encryption key by RID, then decrypted using the same symmetric key by PDM

Fig. 4.1.9: PDM-RID UWB Communication Protocol

Specifications for the behavior implemented during this section are shown in Table 4.1.1.

| Design ID | Specification Description | Referenced Requirement |
|---|---|---|
| D-4.1.0.1.a | The PES™ UWB communication protocol will follow the IEEE 802.15.4-2011 standard. | R-5.2.1.1.a, R-5.2.2.1.a |
| D-4.1.0.2.a | The PES™ UWB communication protocol will implement time-of-flight calculations to perform positioning within 10cm of accuracy. | R-5.2.1.1.a, R-5.2.2.1.a, R-5.3.1.1.a |
| D-4.1.0.3.a | The PES™ UWB communication protocol will implement secure RID authentication. | R-5.3.1.2.a, R-5.3.1.3.a, R-5.3.2.2.a, R-5.3.2.4.a |
| D-4.1.0.4.a | The PES™ UWB communication protocol will use low-power polling to begin location-based communications. | R-5.2.1.1.a, R-5.2.2.1.a, R-5.3.1.1.a, R-5.3.2.4.a |
| D-4.1.0.5.a | The PES™ UWB communication protocol will use a CRC to detect transmission errors at the receiver | R-5.2.1.1.a, R-5.2.2.1.a |
| D-4.1.0.6.a | The PES™ UWB communication protocol will be resilient to lost packets. | R-5.2.1.1.a, R-5.2.2.1.a |
| D-4.1.0.7.a | The PES™ UWB communication protocol will safely time out once out of range. | R-5.2.1.1.a, R-5.2.2.1.a, R-5.3.1.1.a, R-5.3.2.1.a |

Table 4.1.1: UWB Communication Design Specifications

## 4.2 Controller Access Network (CAN) Bus Communication

Similar to UWB, a communication protocol for the CAN bus must be developed. In general, CAN bus communications can be divided into two groups: control messages, which can be done in a single packet like bike state changes, and data messages, which need to be done in an ordered packet stream to ensure data integrity on the receiving end. Figure 4.2.1 shows the format of a standard CAN frame as specified by CAN 2.0A [13]. Note that, in the event that five sequential bits are transmitted of the same polarity, the CAN protocol inserts a *stuff bit* of the opposite polarity to maintain data synchronization.

- The *Arbitration* field is the identifier of the CAN message. If two messages are transmitted simultaneously, arbitration will occur and the message with the ID closer to zero is transmitted.
- The *RTR* bit indicates whether the sender is requesting data with a *Remote Transmission Request*.

- The *IDE* bit indicates whether the message is a standard or extended CAN frame.
- The *R0* bit is reserved to be zero.
- The *DLC* field encodes the length of the data field in bytes.
- The *Data* field includes the data to be transmitted - in Figure 4.1.1, this is just one byte.
- The *CRC* field is the *Cyclic Redundancy Check*, which is used to verify received frame integrity.
- The *ACK* bit indicates whether the frame is an *ACKnowledgement* message.

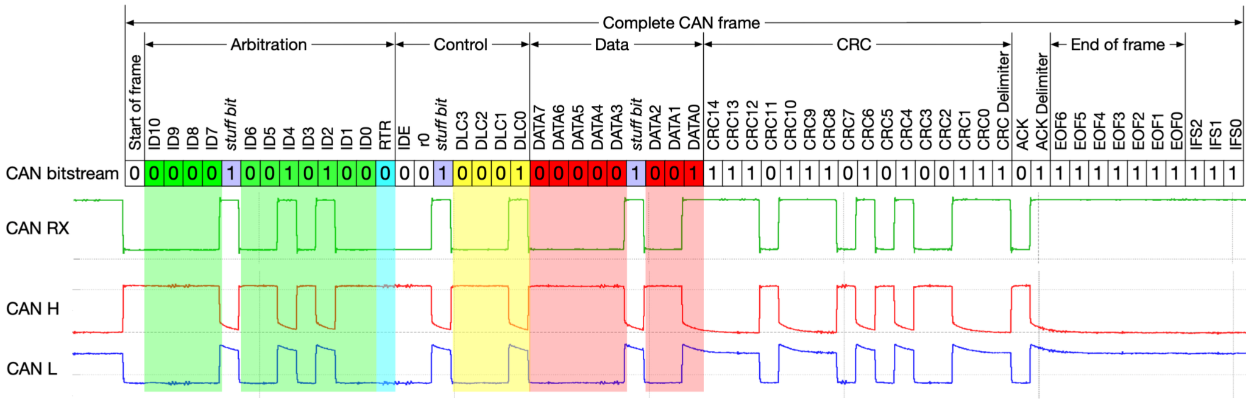


Fig. 4.2.1: Standard CAN Frame Format [14]

### 4.2.1 CAN Control Messages

CAN control messages are the simpler of the two logical CAN message types to implement. To do so, the sender prepares a message with the following fields:

- Arbitration ID: an identifier corresponding to the resource to control. For example, if the PDM is updating the motorcycle's state to 'running', this field would be set to some identifier that corresponds to a MOTORCYCLE_STATE_RUNNING action.
- DLC: set to "1111" since the encrypted data will be 64 bits.
- Data: includes the sender's ID and a rolling code, both encrypted with the shared symmetric key.

When the frame is received, the receiver decrypts the data field with its own symmetric key, then validates the ID of the sender against its internal expected value. Once this is done, provided there were no errors in transit, the receiver issues the action that was requested. If the CAN controller detected an error in transit (eg. CRC mismatch), it sends a predefined CAN error frame that interrupts the CAN bus after which the sender automatically attempts to retransmit their message.

## 4.2.2 CAN Data Messages

Data messages are used to transfer data larger than 64 bits over the CAN bus. In this case, a transmission scheme is needed that can reliably transmit data in order. For this, a simple initialization handshake occurs in which the sender transmits the number of 64-bit data frames it is expecting to send. The receiver can then allocate the memory needed to store the incoming data prior to its arrival.

A communication scheme for CAN data transfer is shown in Figure 4.2.2. Note that, due to the fault tolerance implemented by the CAN controller, ACKs are automatically handled and that bit-level CRC errors will also lead to an automatic retransmit.

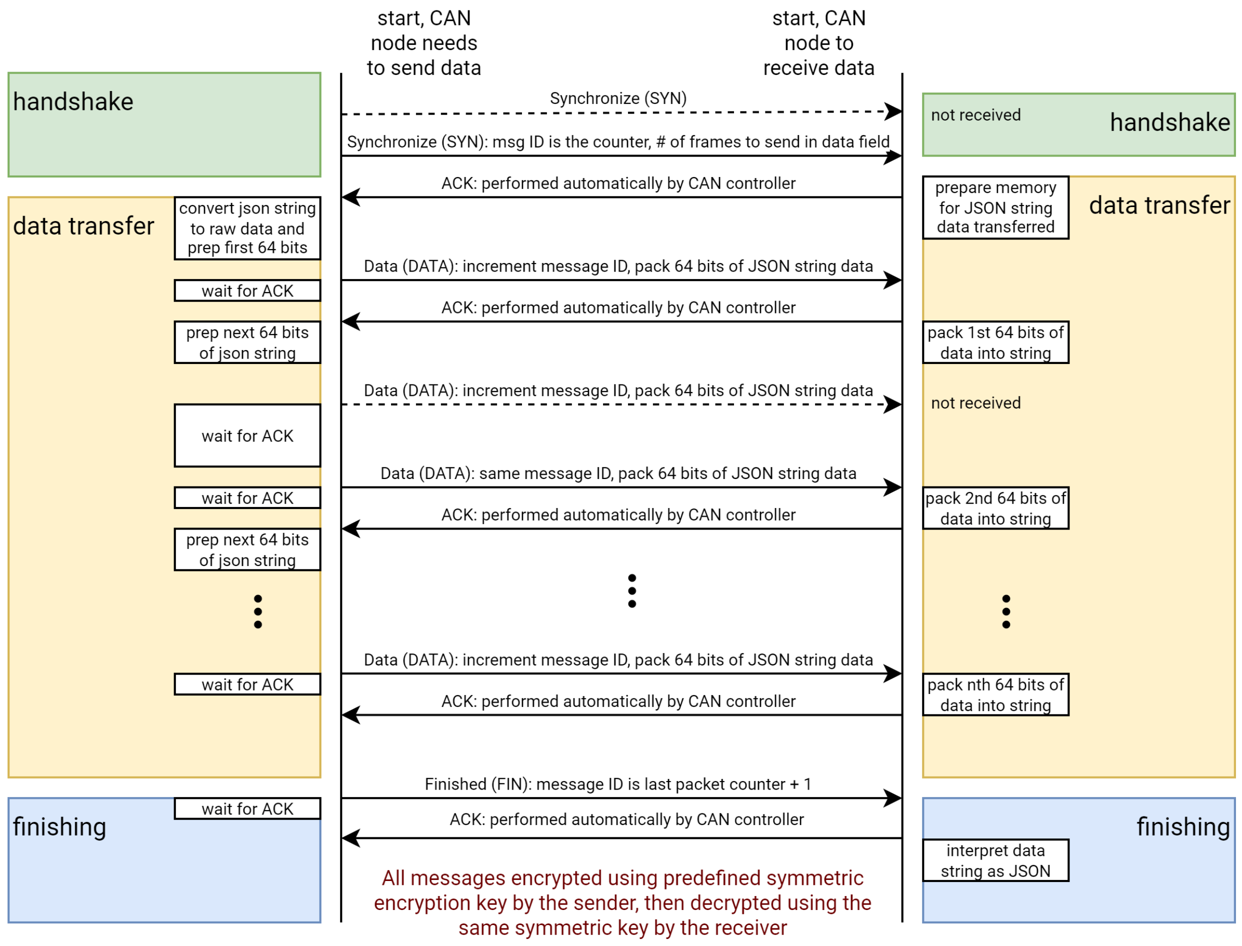

start, CAN node needs to send data
start, CAN node to receive data
handshake
Synchronize (SYN)
not received
handshake
Synchronize (SYN): msg ID is the counter, # of frames to send in data field
ACK: performed automatically by CAN controller
prepare memory for JSON string data transferred
data transfer
convert json string to raw data and prep first 64 bits
data transfer
Data (DATA): increment message ID, pack 64 bits of JSON string data
wait for ACK
ACK: performed automatically by CAN controller
prep next 64 bits of json string
pack 1st 64 bits of data into string
Data (DATA): increment message ID, pack 64 bits of JSON string data
not received
wait for ACK
Data (DATA): same message ID, pack 64 bits of JSON string data
wait for ACK
ACK: performed automatically by CAN controller
pack 2nd 64 bits of data into string
prep next 64 bits of json string
Data (DATA): increment message ID, pack 64 bits of JSON string data
wait for ACK
ACK: performed automatically by CAN controller
pack nth 64 bits of data into string
Finished (FIN): message ID is last packet counter + 1
finishing
wait for ACK
ACK: performed automatically by CAN controller
finishing
interpret data string as JSON
All messages encrypted using predefined symmetric encryption key by the sender, then decrypted using the same symmetric key by the receiver

Fig. 4.2.2: PDM-ECU CAN Data Transfer Protocol

The specifications for CAN behavior between the PDM and ECU for the PES™ are shown in Table 4.2.1.

| **Design ID** | **Specification Description** | **Referenced Requirement** |
|---|---|---|
| D-4.2.2.1.a | The PES™ CAN communication protocol will follow the Bosch CAN2.0a standard. | R-4.2.2.3.a |
| D-4.2.2.2.a | The PES™ CAN communication protocol will implement fixed-length control messages. | R-4.2.2.3.a, R-4.2.2.4.c, R-5.2.2.2.a |
| D-4.2.2.3.a | The PES™ CAN communication protocol will implement variable-length data messages. | R-4.2.2.5.a |

Table 4.2.1: CAN Communication Design Specifications

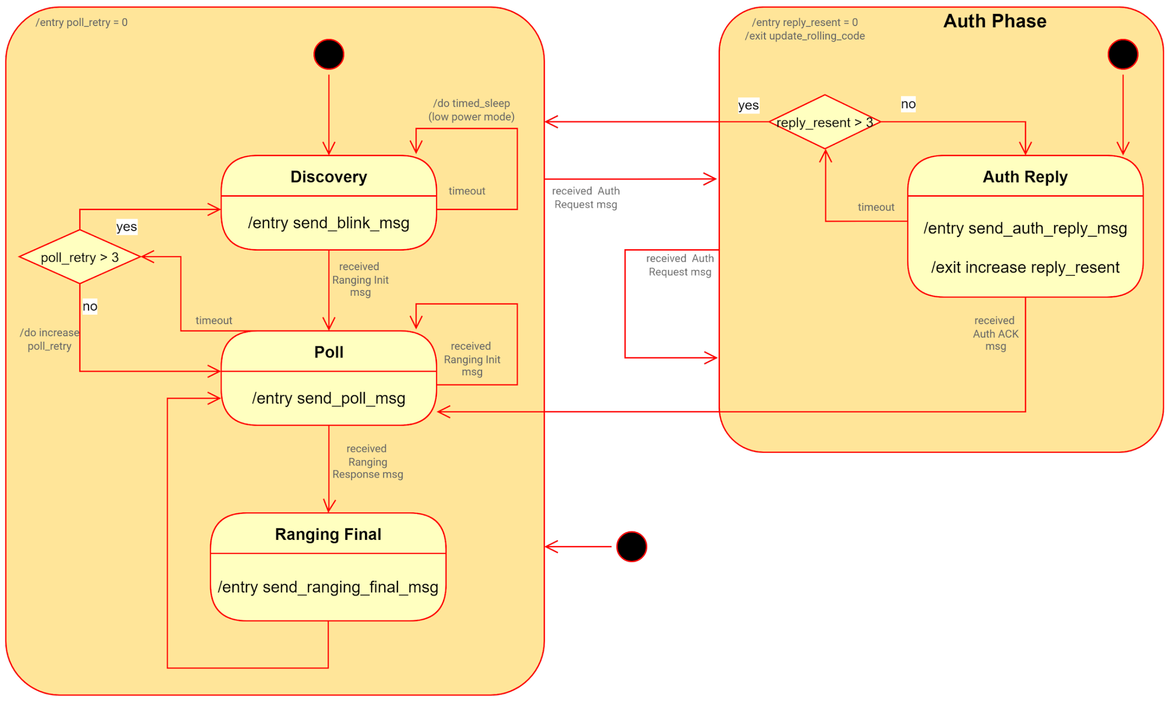
## 5. Software

Software for the PES™ can be split into three types: the Remote Identifier (RID) MCU logic, the Proximity Detection Module MCU logic, and the simulated ECU effectively used as a testbench for the system.

A mechanism based on retries and timeouts was preferred for the RID and PDM software architecture since the number of packets needed for each phase is minimal and both retransmission time and turnaround times (time to go from transmitting to receiving and the other way around) for the DW1000 module are in the order of microseconds [15].

### 5.1 Remote Identifier (RID)

The software for the RID is focused around implementation of the UWB communication protocol described in Section 4.1. As a remote device, it primarily cycles between PDM discovery, positioning, and authentication with the PDM to perform unlocks. The RID software prioritizes authentication over its positioning processes. This enhances the responsiveness of the RID to authentication requests triggered by the PDM when an action is to be performed based on the detected intent, which is specified in Section 5.2. This process is defined in detail in Figure 5.2.1.


/entry poll_retry = 0
/do timed_sleep (low power mode)
Discovery
/entry send_blink_msg
timeout
yes
poll_retry > 3
no
/do increase poll_retry
timeout
received Ranging Init msg
Poll
/entry send_poll_msg
received Ranging Init msg
received Ranging Response msg
Ranging Final
/entry send_ranging_final_msg
received Auth Request msg
received Auth Request msg
/entry reply_resent = 0
/exit update_rolling_code
Auth Phase
yes
reply_resent > 3
no
Auth Reply
/entry send_auth_reply_msg
/exit increase reply_resent
timeout
received Auth ACK msg

Assume all communication is between a PDM and a registered RID for this State Diagram

Fig. 5.1.1: RID State Diagram

The UWB transceiver affords the RID's software the ability to do simple MAC operations. As a result, CRC calculation and checking are performed automatically, a 64-bit MAC address can be created and stored in the transceiver, and automatic message filtering based off of the incoming destination MAC address can be switched on. The RID software uses these features to reduce power consumption on all devices.

Table 5.1.1 contains the specifications necessary for the RID software to implement all required functionality.

| **Design ID** | **Specification Description** | **Referenced Requirement** |
|---|---|---|
| D-5.1.1.1.a | The RID software will regularly transmit blink messages to perform local PDM discovery. | R-5.2.1.1.a, R-5.3.1.1.a |
| D-5.1.1.2.b | The RID software will sleep for a predefined time between blink messages to preserve power. | R-3.4.1.2.b |
| D-5.1.1.3.a | The RID software will enable hardware MAC address filtering to reduce incoming message noise. | R-5.2.1.1.a, R-5.3.1.1.a |
| D-5.1.1.4.a | The RID software will include its full MAC address with each blink message to assist PDM MAC address filtering. | R-5.3.1.3.a |
| D-5.1.1.5.a | The RID software will use scheduled messaging to implement message timestamps for time-of-flight determination. | R-5.2.1.1.a, R-5.3.1.1.a |
| D-5.1.1.6.a | The RID software will use timeouts and limited retry messaging to fall back to discovery when moving out of range. | R-5.1.1.4.b |
| D-5.1.1.7.a | The RID software will authenticate with the PDM when user intent is successfully detected. | R-5.2.1.1.a, R-5.3.1.1.a |

Table 5.1.1: RID Software Design Specifications

## 5.2 Proximity Detection Module (PDM)

The software for the PDM can be generally viewed as a state machine, shown below in Section 5.2.2. This state machine is primarily based around the UWB protocol defined in Section 4.1 with simple callbacks to interface over the CAN bus with the simulated ECU. Before the state diagram is defined, it's important to understand how the PDM will perform RID positioning over UWB.

## 5.2.1 UWB Positioning

Ultra-wideband (UWB) communication can be used to accurately determine the location of a sensor in reference to a second sensor to high precision. This is implemented by using the time of flight of packets sent between the two sensors. This is implemented as described in the rest of this section, which is based off of the APS103 application note for two-way ranging provided by DecaWave [5].

Figure 5.2.1 shows the implementation of *symmetric two-way ranging*, which uses two messages with four timestamps $\mathrm{TX}_1$, $\mathrm{Rx}_1$, $\mathrm{Tx}_2$, and $\mathrm{Rx}_2$, that correspond to the transmission time and received time for the two messages, respectively. Then, by using the initiator's time difference $T_I$ and the responder's time difference $T_R$, the time of flight can be found as

$$\left.\begin{aligned} T_{\text{roundtrip}} &= \mathrm{Rx}_2 - \mathrm{Tx}_1 \\ T_{\text{reply}} &= \mathrm{Tx}_2 - \mathrm{Rx}_1 \end{aligned}\right\} \longrightarrow t_{\text{tof}} = \frac{1}{2}\left(T_{\text{roundtrip}} - T_{\text{reply}}\right) .$$

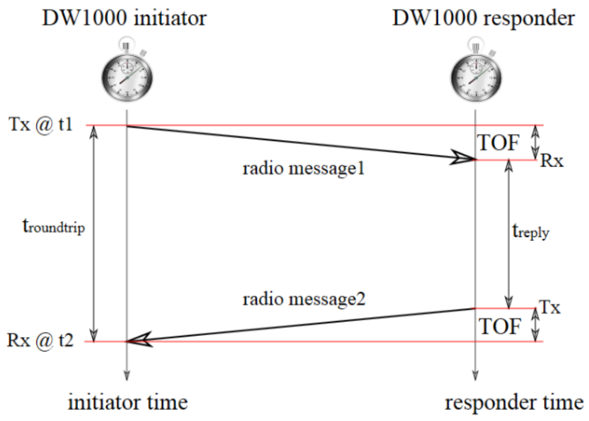


Fig. 5.2.1: Symmetric Two-Way Ranging [5]

An issue with symmetric two-way ranging is that error accumulates due to clock drift and frequency drift [5]. For this reason, *asymmetric two-way ranging* is used to improve performance at the cost of an extra message per transaction. This scheme is shown in Figure 5.2.2. Here, the time of flight for poll/response and for response/final are combined to reduce the effect of measurement error, which gives the total time of flight to be

$$\left.\begin{aligned} T_{\text{round1}} &= \mathrm{Rx}_{\text{resp}} - \mathrm{Tx}_{\text{poll}} \\ T_{\text{reply1}} &= \mathrm{Tx}_{\text{resp}} - \mathrm{Rx}_{\text{poll}} \\ T_{\text{reply2}} &= \mathrm{Tx}_{\text{final}} - \mathrm{Rx}_{\text{resp}} \\ T_{\text{round2}} &= \mathrm{Rx}_{\text{final}} - \mathrm{Tx}_{\text{resp}} \end{aligned}\right\} \longrightarrow t_{\text{tof}} = \frac{T_{\text{round1}}T_{\text{round2}} - T_{\text{reply1}}T_{\text{reply2}}}{T_{\text{round1}} + T_{\text{round2}} + T_{\text{reply1}} + T_{\text{reply2}}} .$$

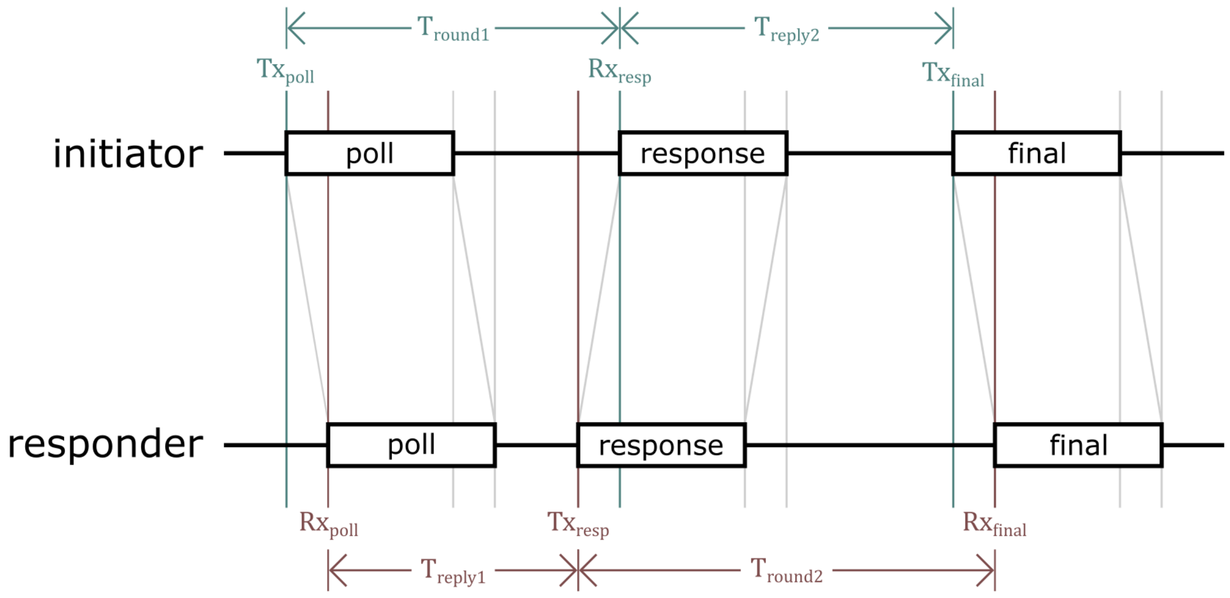


Fig. 5.2.2: Asymmetric Two-Way Ranging

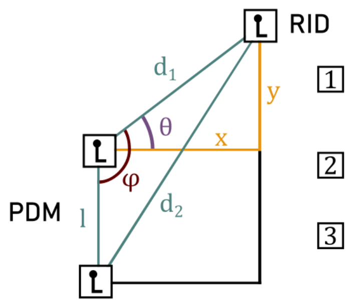
This value for the time of flight will be in seconds. Because UWB is based on RF, the time of flight can be converted to a distance measurement $d$ using the speed of light $c$ as

$$d = c \cdot t_{\text{tof}} \ , \quad c \approx 2.99 \times 10^8 \ \text{m/s} \ .$$

Three antennas can be used to triangulate the position of a device, where two antennas are anchored (the PDM) and one antenna is free to move (the RID). A derivation for the RID position in terms of the distances determined by two antennas attached to the PDM is shown in Figure 5.2.3.

- Equation (1) follows from basic trigonometry
- Equation (2) is the physical definition of θ in radians
- Equation (3) is the Cosine Law solved for the angle φ
- The upper option for (x,y) is found by plugging each equation in, then noting sin(φ-$\pi$/2)=-cos(φ) and that cos(φ-$\pi$/2)=sin(φ)
- The lower option for (x,y) is found by noting that sin(arccos(x))=sqrt(1-$x^2$)

Figure 5.2.3 shows that there are two options to implement (x,y) positioning of the RID. One of these will be chosen depending on which implementation is faster, noting that sin and arccos can be implemented quickly using lookup tables.



$$\boxed{1}\quad x=d_1\cos(\theta) \qquad y=d_1\sin(\theta)$$

$$\boxed{2}\quad \theta=\varphi-\frac{\pi}{2}$$

$$\boxed{3}\quad \varphi=\arccos\left(\frac{d_1^2+l^2-d_2^2}{2d_1 l}\right)$$

$$\left.\begin{aligned} x&=d_1\sin\left(\arccos\left(\frac{d_1^2+l^2-d_2^2}{2d_1 l}\right)\right)\\ y&=-\frac{d_2^2-d_1^2-l^2}{2l}\end{aligned}\right\}$$

or

$$\left.\begin{aligned} x&=d_1\left(1-\frac{(d_1^2+l^2-d_2^2)^2}{4d_1^2l^2}\right)^{1/2}\\ y&=-\frac{d_2^2-d_1^2-l^2}{2l}\end{aligned}\right\}$$

Fig. 5.2.3: UWB Position Triangulation Derivation

To determine the user's intent, the PDM will regularly calculate the position of the RID using its two antennas once it has come into range. These successive position points will be used to compute motion vectors to extrapolate the movement of the user. If enough points fall along a vector that coincides with the PDM, it issues a command to the motorcycle's central ECU to wake up the systems necessary to start the bike. Otherwise, if the user passes a threshold distance after the RID and PDM have begun UWB communication, the PDM will issue a signal to lock the motorcycle.

Table 5.2.1 contains the design specifications needed to implement UWB positioning with the PDM.

| **Design ID** | **Specification Description** | **Referenced Requirement** |
|---|---|---|
| D-5.2.1.1.a | The PDM software will track the position of any registered RID within a predefined threshold distance. | R-5.2.2.1.a, R-5.3.2.1.a |
| D-5.2.1.2.a | The PDM software will issue a "wake-up" signal to the simulated ECU if any registered RID follows a path towards the PDM in range. | R-5.2.2.2.a, R-5.2.2.3.a, R-5.2.2.5.b |
| D-5.2.1.3.a | The PDM will issue a "lock" signal to the simulated ECU if any registered RID exits a predefined threshold distance. | R-5.2.2.10.a |

Table 5.2.1: PDM UWB Positioning Design Specifications

### 5.2.2 PDM Software Model

The software model for the PDM can be separated into two parts: one for UWB processing, and the second for CAN bus interaction. On top of this, a mechanism that transitions from the *Listening* and *Positioning Phase* to the *Authentication Phase* is required for the PDM that performs as efficiently as possible.

Since the PDM triggers bike actions based on positioning data sent by the RID during the *Positioning Phase*, it must securely authenticate the same RID before performing the triggered action, then notify the RID of the successful authentication. The following actions can be triggered by the PDM and notified to the Simulated ECU through CAN:

- Wake-up: disable the alarm and start the booting process for the bike
- Start: enable drive mode when the user presses the start button on the simulated ECU after a Wake-up signal
- Lock: lock the bike and arm the alarm as the user walks away

Figure 5.2.4 shows a finite state machine that defines the implementation of the PDM software model.

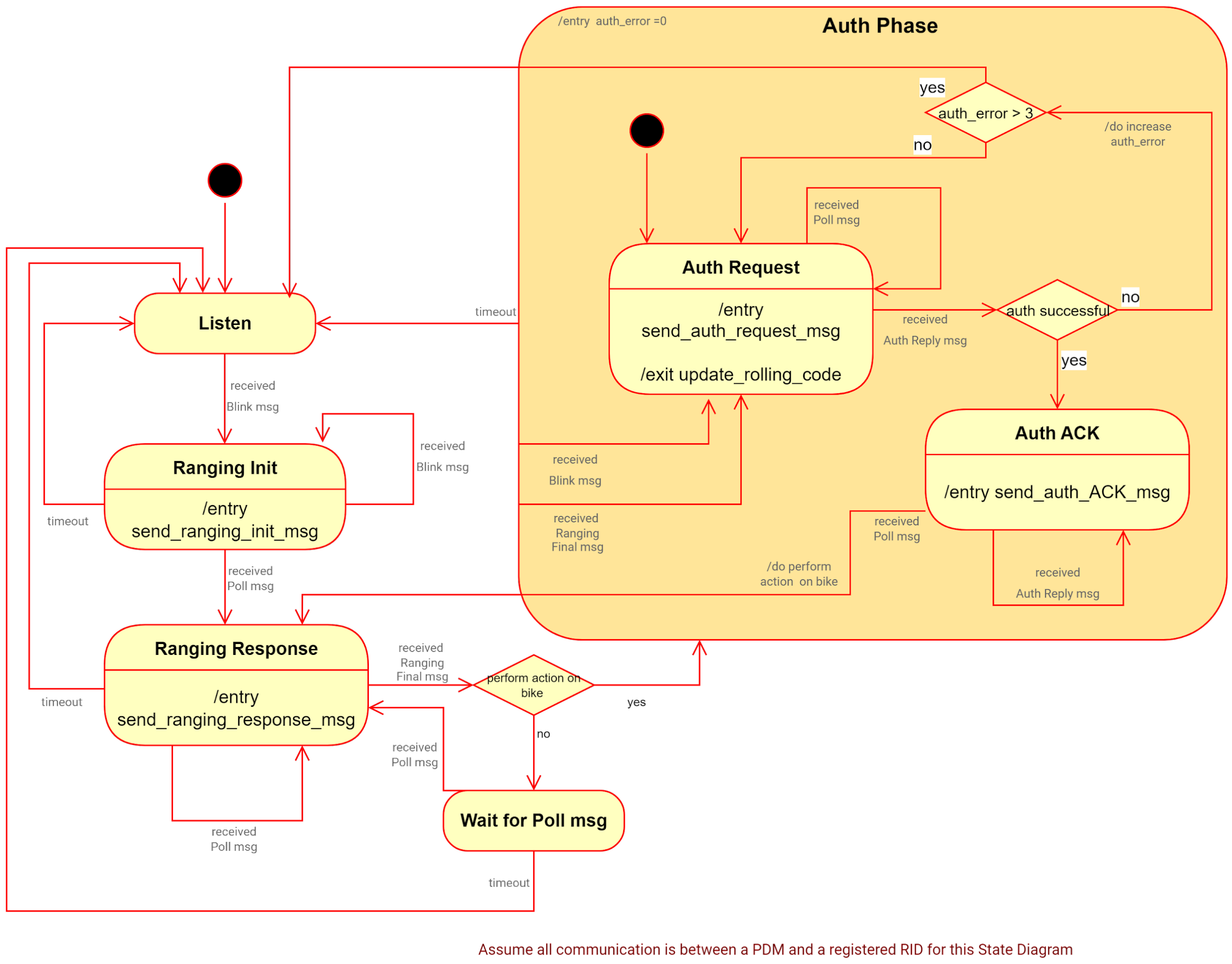


Fig. 5.2.4: PDM UWB State Diagram

Similar to the RID, the UWB transceivers used by the PDM perform basic MAC operations, like CRC calculation and validation, as well as incoming destination MAC address filtering. The PDM software will enable these features to reduce incoming message noise and power consumption.

Table 5.2.2 contains the design specifications needed for the PDM software model to implement ranging with any of its registered RIDs and to implement state communication with the motorcycle ECU.

| **Design ID** | **Specification Description** | **Referenced Requirement** |
|---|---|---|
| D-5.2.1.4.a | The PDM software will initiate position tracking with any registered RID from which it receives a blink message. | R-5.2.2.1.a, R-5.3.2.1.a |
| D-5.2.1.5.a | The PDM software will enable hardware MAC address filtering to reduce message noise and avoid responding to unregistered RIDs. | R-5.2.2.1.a, R-5.3.2.1.a |
| D-5.2.1.6.a | The PDM software will include its full MAC address with responses to blink messages to assist RID MAC address filtering. | R-5.2.2.1.a, R-5.3.2.1.a |
| D-5.2.1.7.a | The PDM software will use timeouts and limited retry messaging to fall back to discovery when all registered RIDs move out of range. | R-5.2.2.10.a |
| D-5.2.1.8.a | The PDM software will not issue any bike actions without first authenticating the RID with which it is communicating. | R-5.2.2.3.a, R-5.2.2.4.a |
| D-5.2.1.9.a | The PDM software will be able to communicate single state changes over the CAN bus. | R-4.2.2.3.a, R-5.2.2.2.a |
| D-5.2.1.10.b | The PDM software will be able to communicate arbitrary data over the CAN bus. | R-4.2.2.5.a, R-4.2.2.4.c |

Table 5.2.2: PDM Software Design Specifications

## 5.3 Simulated ECU

Acting as the vehicle's central ECU, the simulated ECU emulates the local network by sending and receiving CAN messages to and from the PDM. Using a Python library to support CAN, events will be issued through the command-line interface (CLI) to mimic signals sent from the bike. To provide feedback, the simulated ECU will also provide a graphical interface to represent states of the bike. Additionally, the database will be hosted on the simulated ECU device where database entries can be manipulated through the terminal.

Table 5.3.1 designates the required specifications for all functionally needed for the simulated ECU.

| Design ID | Specification Description | Referenced Requirement |
|---|---|---|
| D-5.4.3.1.b | The simulated ECU will display the results using a coloured GUI | R-7.1.2.1.b |
| D-5.4.3.2.b | The simulated ECU will poll for events issued by the PDM and send acknowledgements if successfully received | R-4.2.2.3.a, R-4.2.2.4.c, R-5.2.2.2.a, R-5.2.2.8.b, R-5.2.2.9.a |
| D-5.4.3.3.b | The simulated ECU will update the PDM enrollment table and poll for acknowledgements until successfully sent | R-4.2.2.5.a |
| D-5.4.3.4.b | The simulated ECU will issue a “low power” event to the PDM to shutdown the PDM process. | R-5.1.2.5.b |
| D-5.4.3.5.b | The simulated ECU will create/update/delete entries in the Sqlite database via a suite of Python scripts | R-4.2.2.4.c, R-4.2.2.5.a |

Table 5.3.1: Simulated ECU Software Design Specifications

## 5.4 Database

The PES™ system will require a database to store user information, and public and private keys for both the RID and PDM. The chosen database for this project was a Sqlite database due to its cost, ease of use, and portability (allowing database entries to be passed via CAN). Sqlite is a C-language library that implements a fully featured SQL database in a self-contained database file. This allows team Echo to use the functionality and structure of a SQL database without needing to pay for a fully hosted SQL server. Sqlite is the most widely deployed database engine, used by several web browsers, operation systems, and mobile phones [16]. Sqlite is also natively supported in Python, allowing for easy creation and maintenance of our database on the simulated ECU.

Figure 5.4.1 below shows the Entity-Relationship diagram that will be deployed for the PES™ system. As shown in Figure 5.5.2, the Entity-Relationship diagram will be translated into 3 sqlite tables, “User”, “PDM”, and “RID”.

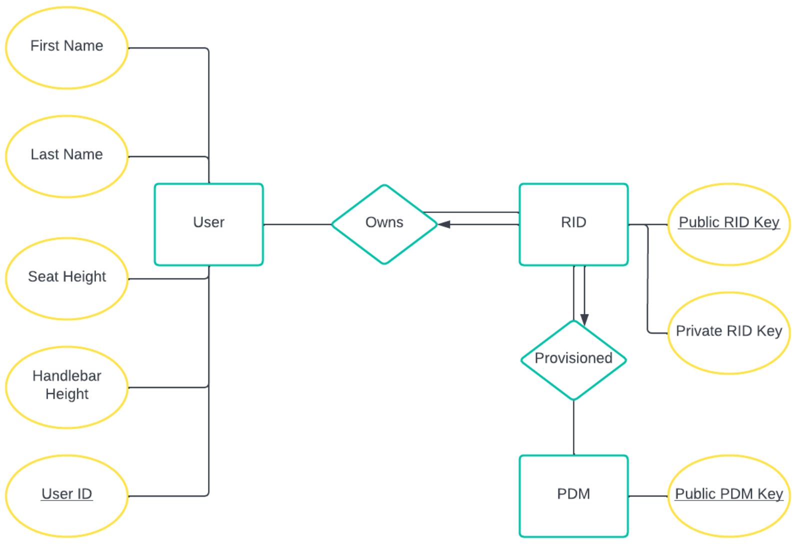


Fig. 5.4.1: Sqlite Database Entity-Relationship Diagram

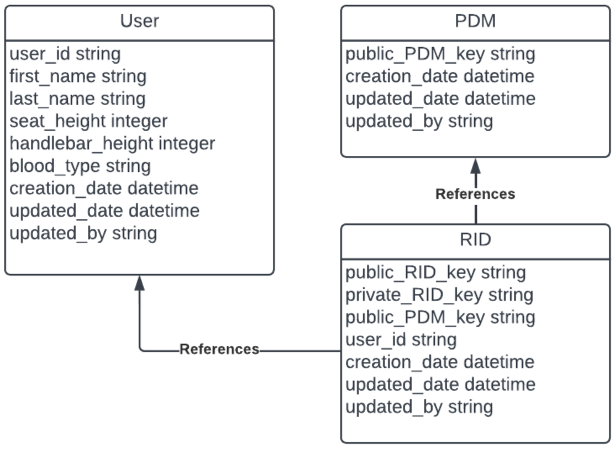


Fig. 5.4.2: Sqlite Database Schema

| **Design ID** | **Specification Description** | **Referenced Requirement** |
|---|---|---|
| D-5.5.0.1.a | The PES™ will use a sqlite database | R-5.1.1.1.a, R-5.1.2.1.a, R-7.1.0.1.b |
| D-5.5.0.2.a | The PES™ database will follow the entity relationship diagram shown in Fig. 4.5.1 | N/A |

Table 5.4.1: Database ECU Software Design Specifications

## 6. Security

Due to its role as a remote keyless entry system, the PES™ has a large focus on security. It needs to be able to uniquely identify its users to grant them access while denying access to malicious actors. For this reason, any common attacks must be considered, as well as potential behavioral considerations. For example, automatically locking and shutting down the bike as the user walks away. Additionally, because the PES™ database will store Personally Identifiable Information (PII), it is important that all database entries be encrypted and plain text storage be avoided.

### 6.1 Ultra-Wideband (UWB) Communication

Because UWB is still a relatively new form of radio communicatication, it does not have much in the way of community support. As shown in Section 4.1, there is no sort of framework or library to lean on to implement a communication protocol - the transceiver only does simple CRC checking and address filtering. As a result, communication security must be carefully considered to prevent bad agents from gaining access to the vehicle. Some of the most common attacks on a remote keyless system are the man-in-the-middle attack, scan attack, replay attack, relay attack, challenge forward prediction attack, and the dictionary attack [17]. The following attack definitions are based on this source.

A *man-in-the-middle attack* occurs when an attacker intercepts packets being sent over a channel, modifies them, and potentially modifies the packets between the two devices. This attack will not work against the PES™, since it uses symmetric encryption for any sensitive packets for CAN and UWB.

A *replay attack* is implemented by an attacker recording the raw data of packets sent, then re-sending these packets in an attempt to repeat that action. For example, an attacker could try to replay the packet that says “start the motorcycle”. To deal with this, during the UWB authentication phase, the RID sends a *rolling code* that is incremented on each authentication packet. The PDM then subtracts its own rolling code value from the decrypted value - if it’s negative or larger than a predefined range, then this is likely a replay attack and it drops the packet. If it’s within the expected range, the packet came from a real registered RID and it proceeds with starting or locking the bike.

A *scan attack* works by rapidly sending successive packets over a network to try to determine identifiers, counters, or other packet fields used for secure transfer. In the proposed UWB communication protocol (Section 4.1), the relevant field to scan for would be the rolling code, which is used to ensure that each secure packet is different from one another. However, the rolling code is encrypted by the RID private key found both in the RID and in the PDM.

A *relay attack* is when an attacker uses an external device to bridge a connection between the vehicle and keyfob. For example, on a typical remote keyless entry system, this would mean tricking the keyfob into thinking that the vehicle is nearby with an external device, then relaying the valid “unlock” packet to the vehicle, despite the owner not being nearby. This does not work with UWB, because the unlock mechanism depends on the time-of-flight information, so the PDM will accurately recognize that the RID is too far away for a successful unlock since the time of flight of the relayed message will still be accurate to perform the distance calculations.

A *challenge forward prediction* attack is when an attacker observes consecutive valid and successful packets sent over a communication channel, then tries to predict the format of the next successful packet and send it to the destination. This doesn’t work, since the rolling code is encrypted by an AES-128 symmetric key by the RID,  each successive authentication packet will look like random noise.

Finally, a *dictionary attack* is when an attacker takes a well-known set of challenge-response pairs and tries each one in an attempt to gain access to the secured device. This won’t work with the system since, similar to the challenge forward prediction attack, each authentication packet will look sufficiently random. Since it’s using a constantly-increasing rolling code, there will be no pattern of responses to the authentication request, so no dictionary of responses will work.

The specifications that the PES™ will use to secure its UWB communications are shown in Table 6.1.1.

| **Design ID** | **Specification Description** | **Referenced Requirement** |
|---|---|---|
| D-6.1.0.1.a | All PES™ UWB authentication will use symmetric AES-128 encryption. | R-4.1.1.2.a, R-4.1.1.3.a, R-4.2.2.2.a, R-5.3.1.2.a, R-5.3.2.2.a, R-7.2.2.1.a, R-7.2.2.3.a, |
| D-6.1.0.2.a | All PES™ UWB encrypted messaging will use rolling codes to eliminate the risk of replay attacks. | R-5.2.2.3.a, R-5.2.2.4.a |
| D-6.1.0.3.a | No PES™ UWB communications will ever transmit any secret keys. | R-7.2.2.3.a |
| D-6.1.0.4.b | The PES™ will provide an interface for manufacturers to set and lock down their own UWB encryption keys at manufacturing time. | R-4.1.1.2.a, R-4.1.2.2.a, R-7.2.2.3.a |

Table 6.1.1: UWB Communication Security Design Specifications

## 6.2 Controller Access Network (CAN) Communication

Like UWB, CAN is essentially a manually-organized transfer of single bits over a physical medium. As a result, careful security considerations must be taken into account to limit the impact of potential attackers. Because of this, CAN communication is also implemented using symmetric AES-128 encryption of data using rolling codes. Because of its similarity to the encryption scheme used for UWB, any common attacks are handled in the same way as described in Section 6.1.

The specifications that the PES™ will use to secure its CAN communications are shown in Table 6.1.2.

| Design ID | Specification Description | Referenced Requirement |
|---|---|---|
| D-6.2.0.1.a | All PES™ CAN messaging will use symmetric AES-128 encryption. | R-4.2.2.2.a, R-5.3.2.3.a, R-7.2.2.3.a |
| D-6.2.0.2.a | All PES™ CAN encrypted messaging will use rolling codes to eliminate the risk of replay attacks. | R-5.2.2.3.a, R-5.2.2.4.a |
| D-6.2.0.3.a | No PES™ CAN communications will ever transmit any secret keys. | R-7.2.2.3.a |
| D-6.1.0.4.b | The PES™ will provide an interface for manufacturers to set and lock down their own CAN encryption keys at manufacturing time. | R-4.1.2.2.a, R-7.2.2.3.a |

Table 6.2.1: CAN Communication Security Design Specifications

## 6.3 Database

A database will be used in conjunction with the simulated ECU to insert, update, and delete user data within the PES™ system. Because the database will store PII as well as user access keys (used to unlock the motorcycle), it is pivotal that the entries stored in the database are encrypted via an industry standard encryption algorithm. The entries stored within the PES™ database will be encrypted using the Advanced Encryption Standard (AES) 256 bit block cipher encryption method. The particular encryption mode used for encryption will be Galois/Counter Mode (GCM) as this allows a user to verify that a decrypted payload has not been tampered with. The GCM algorithm was also selected as it is patent-free.

The AES-256 GCM encryption algorithm uses a single password, which will be stored on the simulated ECU, in conjunction with a randomly generated salt (used to hash the data) to generate a private encryption key to encrypt the entries in the database. The AES-256 GCM encryption algorithm also produces a nonce and tag byte sequence used for decrypting and verifying the decrypted payload, respectively. The following figures display the workflow associated with encrypting and decrypting payloads with this algorithm. The boxes in the large yellow rectangle (Salt, Nonce, Encrypted data, and Tag) are the binary output that will be stored in each entry of the database table.

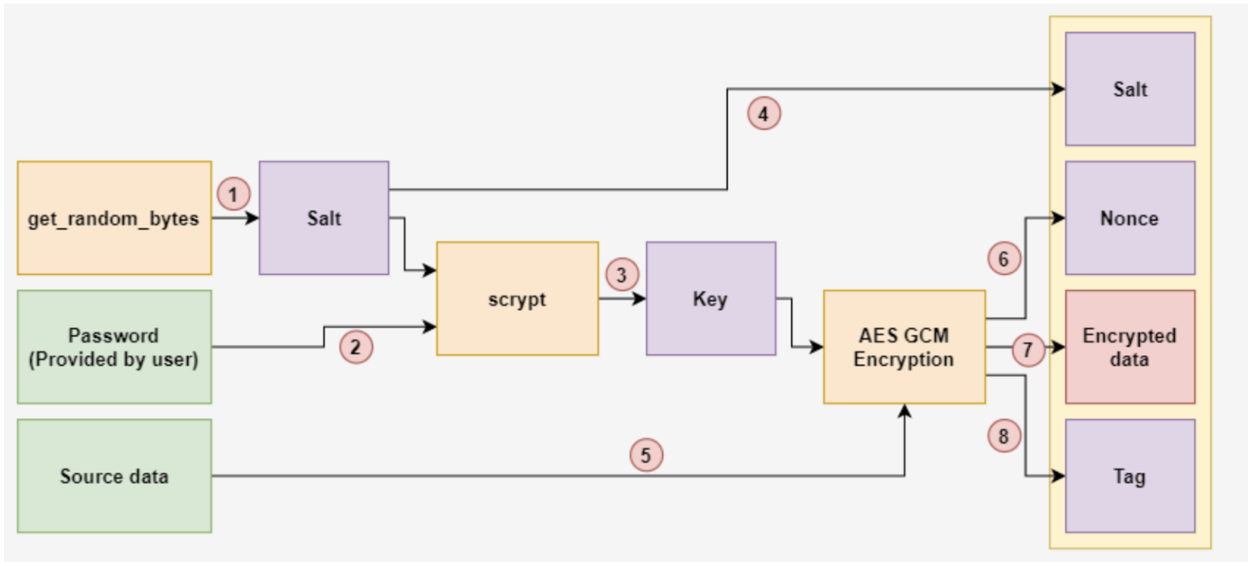


Fig. 6.3.1: AES-256 GCM Encryption Workflow [18]

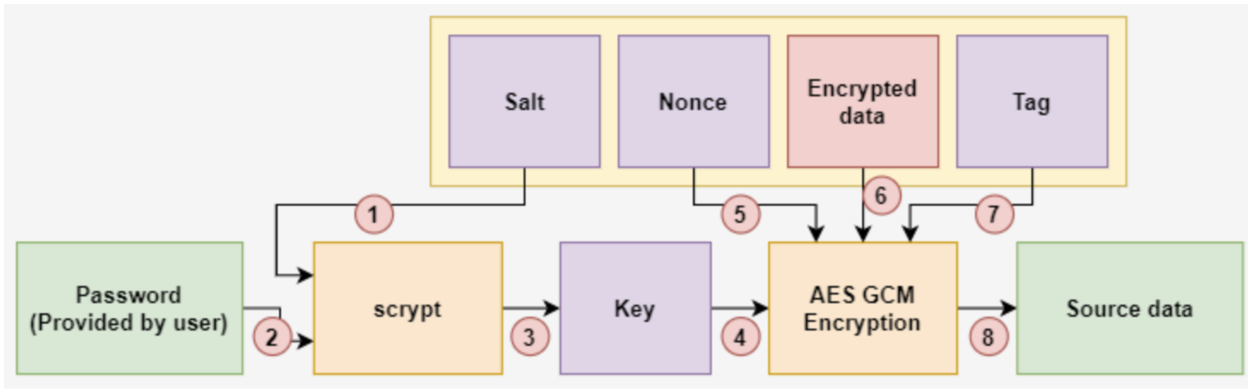


Fig. 6.3.2: AES-256 GCM Decryption Workflow [18]

| Design ID | Specification Description | Referenced Requirement |
|---|---|---|
| D-6.1.0.1.a | The PES™ database will have encrypted entries using an AES-256 GCM encryption algorithm | R-7.2.2.3.a |
| D-6.1.0.2.a | The simulated ECU will contain a suite of python scripts to encrypt and decrypt database entries | N/A |
| D-6.1.0.3.a | The private key for encryption will be stored in a secure location | N/A |
| D-6.1.0.4.a | Decrypted entries will be inspected for tampering | R-7.2.2.1.a |

Table 6.3.1: Database Security Design Specifications

## 7. Physical Design

To protect the internal components of the PES™, a casing is designed to keep the system operational while under the stress of outdoor environment and everyday use. This includes its standards of weight, sizing, mechanical durability, waterproofing, and windproofing.

### 7.1 Casing

The casings will safely secure the PES™ while protecting the hardware from outdoor elements. The design of the PDM and RID will consider portability to remain unobtrusive to the user's experience. PLA filament will be used as 3D printing material for its strength and accuracy for forming a durable case. Figure 7.1.1 and Figure 7.1.2 show mock-ups for the design of the cases.

| **Design ID** | **Specification Description** | **Referenced Requirement** |
|---|---|---|
| D-7.1.1.1.b | The RID casing will be 3D printed using PLA filament | R-3.1.1.1.b<br>R-3.2.1.1.b<br>R-3.2.1.2.b<br>R-9.2.1.1.b |
| D-7.1.2.2.b | The PDM unit casing will be 3D printed using PLA filament | R-3.1.2.1.b<br>R-3.2.2.1.b<br>R-3.2.2.1.b<br>R-9.2.2.1.b<br>R-9.2.2.2.b<br>R-9.2.2.3.b |
| D-7.1.0.3.b | The RID and PDM will be resistant to rain, dust, and wind | R-3.3.1.1.b<br>R-3.3.1.2.b<br>R-3.3.1.2.b<br>R-3.3.1.4.b<br>R-3.3.1.5.b |
| D-7.1.2.4.b | The PDM casing will contain steel mounting brackets that can adhere to irregular surfaces | R-3.5.2.1.b<br>R-3.5.2.2.b<br>R-3.5.2.3.b |

Table 7.1.1: Casing Design Specifications

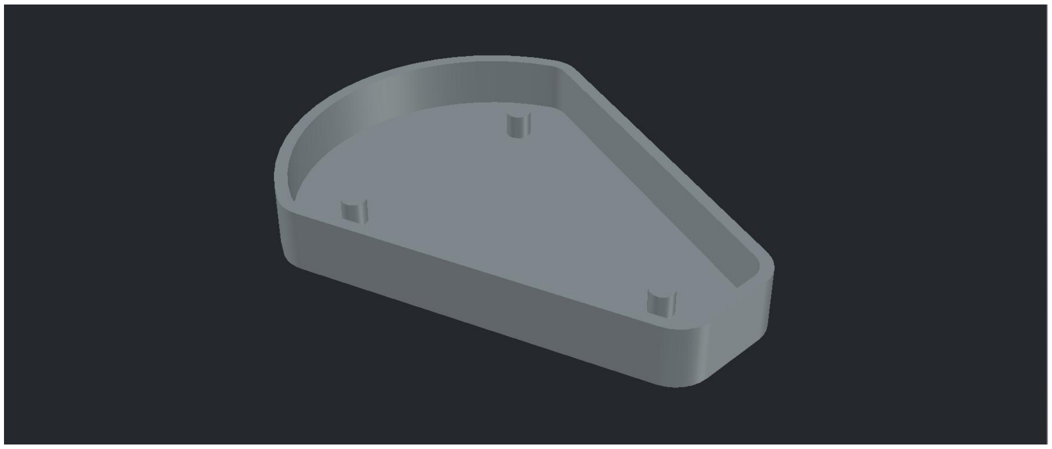


Fig. 7.1.1: RID Casing AutoCAD Model

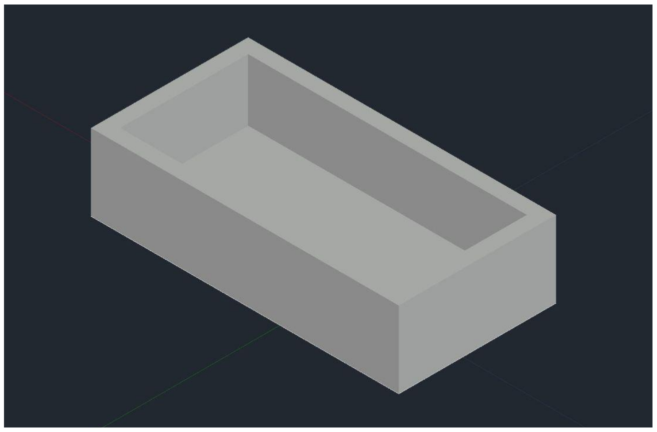


Fig. 7.1.2: PDM Casing AutoCAD Model

## 7.2 Packaging

For the PES™ to find success on the market, it is important that it remains affordable for our target demographic as well as financially attainable to manufacture. It is also paramount that the PES™ maintains the respect of our clientele; by delivering a long-lasting and secure product to the end-user. The packaging specifications are outlined below.

| **Design ID** | **Specification Description** | **Referenced Requirement** |
|---|---|---|
| D-7.2.0.1.c | Individual PEM™ packages will be enclosed in a styrofoam lined box | R-6.3.0.1.c |
| D-7.2.0.2.c | Individual PED™ packages will contain the Echo logo and branding | R-6.3.0.2.c |

Table 7.2.1: Packaging Design Specifications

## Conclusion

The Proximity Entrance System™ is built upon a complex combination of hardware, software, and communication protocols that come together to become as transparent as possible to the user. For the proof of concept, the two discrete components of the PES™ - the PDM and RID - will be built on USB-powered microcontroller development boards connected with the necessary ICs to demonstrate the full functionality of the end-to-end intent detection system.

For the RID component, an STM32 NUCLEO-L432KC development board will supply a 3.3V connection to the DecaWave DWS1000 development board, which powers a DWM1000 UWB chip. They will be connected together over an SPI bus to facilitate raw binary data transfer over the UWB connection. This UWB transmission is what connects the RID to the PDM, and the two communicate securely over a custom-designed communication protocol with symmetric encryption.

On the PDM side, an identical 3.3V connection is used to supply a pair of DWS1000 development boards, which comes from an STM32 NUCLEO-F303RE development board used to implement the PDM. The PDM uses timestamps measured by a synchronized packet exchange between the PDM and RID to calculate the time-of-flight of the packets. This time of flight is used to accurately determine the position of the RID. Once it has determined the user is intending to start the bike, it issues a request to the simulated ECU over its CAN bus, which is implemented by its MCP2551 CAN transceiver. This component is powered by a 5V connection from the microcontroller. The CAN bus also uses its own custom communication protocol that, like the UWB transmission, is protected by a separate set of symmetric encryption keys.

As the “wake-up” request passes through the CAN bus, it is received on the ECU side by the Seeed Studio CAN to USB adapter that contains both a CAN transceiver and controller, which translates the differential voltage signal into a serial message that the simulated ECU can understand. This simulated ECU then displays the change in its user interface, which is matched by the LEDs and speaker reflecting the successful unlock on the RID, which has been notified by the PDM of the successful action. The simulated ECU also facilitates connection to a database that stores user profiles for motorcycle manufacturers to create a unique per-user experience.

## References

[1] "Electric Bike Market", Next Move Strategy Consulting, Tinsukia, Assam, India, Jul. 2021. [Online]. Available: https://www.nextmsc.com/report/electric-bike-market. [Accessed: 12-Jul-2022].

[2] "Nucleo-L432KC," *armMBED*. [Online]. Available: https://os.mbed.com/platforms/ST-Nucleo-L432KC/. [Accessed: 12-Jul-2022].

[3] "When does the key fob battery need replacing?", Testing Autos. [Online]. Available: https://www.testingautos.com/car_care/key-fob-battery.html. [Accessed: 12-Jul-2022]].

[4] "STM32 nucleo-F303RE," *RIOT*. [Online]. Available: https://doc.riot-os.org/group__boards__nucleo-f303re.html. [Accessed: 12-Jul-2022].

[5] "DWM1000 Product Data Sheet," *Qorvo*. [Online]. Available: https://www.qorvo.com/products/p/DWM1000#documents. [Accessed: 12-Jul-2022].

[6] "DWS1000 Qorvo," *Mouser Electronics*. [Online]. Available: https://www.mouser.ca/ProductDetail/Qorvo/DWS1000?qs=TiOZkKH1s2Q1L44eotOGgw%3D%3D. [Accessed: 12-Jul-2022].

[7] CAN bus," *Wikipedia*, 29-Jun-2022. [Online]. Available: https://en.wikipedia.org/wiki/CAN_bus. [Accessed: 12-Jul-2022].

[8] Anonymous, "USB to Can Analyzer Adapter with USB cable," *Seeed Studio*. [Online]. Available: https://www.seeedstudio.com/usb-can-analyzer-p-2888.html. [Accessed: 12-Jul-2022].

[9] "SN65HVD230 can board," *Waveshare*. [Online]. Available: https://www.waveshare.com/sn65hvd230-can-board.htm. [Accessed: 12-Jul-2022].

[10] "Microchip." [Online]. Available: https://www.microchip.com/en-us/product/MCP2551. [Accessed: 12-Jul-2022].

[11] "DecaWave APS013," *Qorvo*. [Online]. Available: https://www.decawave.com/aps013/. [Accessed: 14-Jul-2022].

[12] "IEEE SA - IEEE standard for local and metropolitan area networks--part 15.4: Low-rate wireless personal area networks (LR-WPANs)," IEEE Standards Association. [Online]. Available: https://standards.ieee.org/ieee/802.15.4/5050/#:~:text=This%20standard%20defines%20the%20protocol%20and%20compatible%20interconnection%20for%20data,personal%20area%20network%20(WPAN). [Accessed: 14-Jul-2022].

[13] "Can specification 2.0: Protocol and implementations," SAE MOBILUS. [Online]. Available: https://saemobilus.sae.org/content/921603/. [Accessed: 14-Jul-2022].

[14] Ken, "The canframe.py tool," CANIS Automotive Labs, 25-Oct-2020. [Online]. Available: https://kentindell.github.io/2020/01/03/canframe_py_tool/. [Accessed: 14-Jul-2022].

[15] "DW100 Product Data Sheet," *Qorvo*. [Online]. Available: https://www.qorvo.com/products/p/DW1000#documents. [Accessed: 12-Jul-2022].

[16] "What is SQLite?," Sqlite. [Online]. Available: https://www.sqlite.org/index.html. [Accessed: 12-Jul-2022].

[17] A. I. Albrabady, and S. M. Mahmud, "Analysis of attacks against the security of keyless-entry systems for vehicles and suggestions for improved designs," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 8, pp. 41-50, Jan. 2005. Accessed: Jul. 13, 2022. doi: 10.1109/TVT.2004.838829. [Online]. Available: https://ieeexplore.ieee.org/document/1386610

[18] B. Vollebregt, "Python GCM Encryption Tutorial," *Nitratine*. [Online]. Available: https://nitratine.net/blog/post/python-gcm-encryption-tutorial/. [Accessed: 12-Jul-2022].

[19] "ATA8350-7MQW Microchip," Mouser Electronics. [Online]. Available: https://www.mouser.ca/ProductDetail/Microchip-Technology-Atmel/ATA8350-7MQW?qs=DRkmTr78QARUtCiXY4Fn%2FQ%3D%3D. [Accessed: 14-Jul-2022].

[20] "NCJ29D5," *Mouser Electronics*. [Online]. Available: https://www.mouser.ca/ProductDetail/NXP-Semiconductors/NCJ29D5BHN-00200Y?qs=vmHwEFxEFR8d8wg4m%2FWbHQ%3D%3D. [Accessed: 14-Jul-2022].

[21] "DW1000," *Mouser Electronics*. [Online]. Available: https://www.mouser.ca/ProductDetail/Qorvo/DW1000-I-TR13?qs=TiOZkKH1s2SHqHtNDSAfuQ%3D%3D. [Accessed: 14-Jul-2022].

[22] "DW3110," *Mouser Electronics*. [Online]. Available: https://www.mouser.ca/ProductDetail/Qorvo/DW3110TR13?qs=hWgE7mdIu5Tls8vsL0LU5w%3D%3D. [Accessed: 14-Jul-2022].

[23] "STM32F103C8," *STMicroelectronics*. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html#documentation. [Accessed: 14-Jul-2022].

[24] "STM32L432KC," *armMBED*. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32l432kc.html#documentation [Accessed: 12-Jul-2022]

[25] "STM32F303," *STMicroelectronics*. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f303.html. [Accessed: 14-Jul-2022].

[26] "Arduino® Uno R3." [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf. [Accessed: 14-Jul-2022].

[27] "MCP2551-I/SN", *Mouser Electronics*. [Online]. Available: https://www.mouser.ca/ProductDetail/Microchip-Technology/MCP2551-I-SN?qs=9y3LFqDLL8L5zFfqqxdOHg%3D%3D. [Accessed: 10-Jul-2022].

[28] "TJA1050T/CM,118", Mouser Electronics. [Online]. Available: https://www.mouser.ca/ProductDetail/NXP-Semiconductors/TJA1050T-CM118?qs=5XOdhvmYM0NFFspw0VP5sA%3D%3D [Accessed: 10-Jul-2022].

[29] "TLT9251VLEXUMA1", *Mouser Electronics*. [Online]. Available: https://www.mouser.com/ProductDetail/Infineon-Technologies/TLT9251VLEXUMA1?qs=W%2FMpXkg%252BdQ5zGtp9sfvtfA%3D%3D. [Accessed: 10-Jul-2022].

[30] "TCAN1051HVD", Mouser Electronics. [Online]. Available: https://www.mouser.com/ProductDetail/Texas-Instruments/TCAN1051HVD?qs=dSktwyqbRaXvzo3KnBk82Q%3D%3D. [Accessed: 10-Jul-2022].

[31] Seed Studio CAN Analyzer, Seed Studio. [Online]. Available: https://www.seeedstudio.com/usb-can-analyzer-p-2888.html [Accessed: 10-Jul-2022].

[32] "USB-to-CAN V2", *Ixxat*. [Online]. Available: https://www.ixxat.com/products/products-industrial/can-interfaces/usb-can-interfaces/usb-to-can-v2-professional?ordercode=1.01.0281.12001 [Accessed: 10-Jul-2022].

[33] "Kvaser Leaf Light HS v2", *KVASER*. [Online]. Available: https://www.kvaser.com/product/kvaser-leaf-light-hs-v2/ [Accessed: 10-Jul-2022].

[34] "CAN USB Adapter (PCAN-USB)", *gridconnect*. [Online]. Available: https://www.gridconnect.com/products/can-usb-adapter-pcan-usb. [Accessed: 10-Jul-2022].

[35] "An open-source USB to can adapter," CANable. [Online]. Available: https://canable.io/. [Accessed: 14-Jul-2022].

[36] "Top 20 databases to use in 2022: A review of the latest technologies and Tools," *Brainvire*, 06-Jan-2022. [Online]. Available: https://www.brainvire.com/blog/top-20-databases-to-use/. [Accessed: 12-Jul-2022].

[37] S. Turki, "8 Best Database Software For Small Business," *StartUpLift*, 19-Jun-2020. [Online]. Available: https://startuplift.com/best-database-software/. [Accessed: 12-Jul-2022].

[38] "Best relational databases for small businesses in 2022," *G2*. [Online]. Available: https://www.g2.com/categories/relational-databases/small-business. [Accessed: 13-Jul-2022].

[39] "Pricing," *MongoDB*. [Online]. Available: https://www.mongodb.com/pricing. [Accessed: 12-Jul-2022].

[40] D. A Norman, *The design of everyday things*. New York: Basic Books, 2013.

[41] *Road vehicles — Ergonomics aspects of transport information and control systems — Human machine interface specifications for keyless ignition systems,* ISO 21956:2019, November 2019. [Online]. Available: https://www.iso.org/standard/72290.html [Accessed: 12-Jul-2022].

[42] *Road vehicles — Ergonomics of human-system interaction — Part 161: Guidance on visual user-interface elements,* ISO 9241-161:2016, February 2016. [Online]. Available: https://www.iso.org/standard/60476.html [Accessed: 12-Jul-2022].

[43] *Systems and software engineering — Software life cycle processes*, ISO/IEC/IEEE 12207:2017, June 2022. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:12207:ed-1:v1:en. [Accessed: 12-Jul-2022].

[44] *Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling*, ISO 11898-1:2015, December 2015. [Online]. Available: https://www.iso.org/standard/63648.html [Accessed: 12-Jul-2022].

[45] *Information technology — Telecommunications and information exchange between systems — High-rate ultra-wideband PHY and MAC standard*, ISO/IEC 26907:2009, November 2009. [Online]. Available: https://www.iso.org/standard/53426.html. [Accessed: 12-Jul-2022].

*[46] IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, Feb 2011. [Online]. Available: https://standards.ieee.org/ieee/802.15.4/5050/ [Accessed: 10-Jul-2022].

[47] M. Sjoerdsma. (2022). Lecture 5 [PowerPoint slides]. Available: https://canvas.sfu.ca/courses/69950/files/folder/Powerpoints?preview=19281624

# Appendix A: Test Plan

## A.1 Introduction

### A.1.1 Test Purpose

The PES™ test plan provides detailed steps for acceptance testing for the proof-of-concept demo, ensuring that all design specifications and requirements for the prototype are met.

### A.1.2 Test Coverage

The test plan presented is associated solely with the proof-of-concept prototype. The tests outlined cover various hardware and software tests belonging to the RID, PDM, Simulated ECU, as well as overall PES™ security. The tests cover a priority range from Low to Medium to High, this allows Echo to classify which tests are critical to the overall success of the product, and which carry a lower importance.

### A.1.3 Test Methods

The testing will take place in Lab 1 at Simon Fraser University. The members of team Echo will bring all sub components of the PES™ system and verify the tests outlined below.

### A.1.4 Test Responsibilities

The testing will be performed by all members of Echo following the test outlines presented below.

## A.2 Remote Identifier (RID) Testing

| | |
|---|---|
| **Test Name:** RID Approaches PDM Test | **Priority (Low/Medium/High):** High |
| **Test Description:** RID follows a motion trajectory that describes the intent of the user to ride the motorcycle | |
| **Acceptance Criteria:** System is issued a “wake-up” signal | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** User Starts Motorcycle | **Priority (Low/Medium/High):** High |
| **Test Description:** RID is detected within a close range and the user presses the start button | |
| **Acceptance Criteria:** System is issued a “start” signal | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** Backup Button Test | **Priority (Low/Medium/High):** Low |
| **Test Description:** Simulated a software intent detection failure, RID backup activation button allows the user to unlock/lock the system | |
| **Acceptance Criteria:** System unlocks | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** Registered RID Test | **Priority (Low/Medium/High):** High |
| **Test Description:** Bring registered RID to PDM with expected user intent | |
| **Acceptance Criteria:** System unlocks | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** Unregistered RID Test | **Priority (Low/Medium/High):** High |
| **Test Description:** RID registered with a different system enters range | |
| **Acceptance Criteria:** System remains locked | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** RID Out of Range Test | **Priority (Low/Medium/High):** High |
| **Test Description:** RID goes out of range and comes back in range and it is still able to communicate with PDM | |
| **Acceptance Criteria:** Positioning phase is being performed between RID and PDM | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

## A.3 Proximity Detection Module (PDM) Testing

| | |
|---|---|
| **Test Name:** PDM Range Test | **Priority (Low/Medium/High):** High |
| **Test Description:** Remove registered RID from PDM range after system was unlocked | |
| **Acceptance Criteria:** System locks | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** PDM Intent Test | **Priority (Low/Medium/High):** High |
| **Test Description:** Bring RID to PDM with no user intent (e.g. passing by) | |
| **Acceptance Criteria:** System remains locked | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** Multiple Registered RIDs Test | **Priority (Low/Medium/High):** High |
| **Test Description:** Two unique registered RIDs (A, B) approach the PDM; RID A meets intent requirements and RID B does not | |
| **Acceptance Criteria:** System unlocks and recognizes user attached to RID A | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** Multiple Registered RIDs Test 2 | **Priority (Low/Medium/High):** Medium |
| **Test Description:** Two unique registered RIDs (A, B) approach the PDM; RID A passes intent requirements, then RID B passes intent requirements | |
| **Acceptance Criteria:** System unlocks and recognizes user attached to RID A (the first RID to pass intent requirements) | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

## A.4 Simulated ECU Testing

| | |
|---|---|
| **Test Name:** ECU Wake Up Display Test | **Priority (Low/Medium/High):** High |
| **Test Description:** Simulated ECU receives wakeup signal from PDM | |
| **Acceptance Criteria:** Simulated ECU displays that a wakeup signal was received | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** ECU Start Signal Display Test | **Priority (Low/Medium/High):** High |
| **Test Description:** Simulated ECU receives start signal from PDM | |
| **Acceptance Criteria:** Simulated ECU displays that a start signal was received | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** ECU Lock Signal Display Test | **Priority (Low/Medium/High):** High |
| **Test Description:** Simulated ECU receives lock signal from PDM | |
| **Acceptance Criteria:** Simulated ECU displays that a lock signal was received | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** ECU Query Test | **Priority (Low/Medium/High):** High |
| **Test Description:** Simulated ECU receives a query from PDM for motorcycle state information | |
| **Acceptance Criteria:** Simulated ECU sends motorcycle state information to PDM | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** ECU Update User Test | **Priority (Low/Medium/High):** Medium |
| **Test Description:** Simulated ECU receives a request to update a User in the database | |
| **Acceptance Criteria:** Simulated ECU updates a User's information in the database | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** ECU Update Registered RID Test | **Priority (Low/Medium/High):** Medium |
| **Test Description:** Simulated ECU receives a request to add/remove an RID from the registered RID list | |
| **Acceptance Criteria:** Simulated ECU queries DB, receives new registered RID list and sends it through CAN | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** ECU Delete Entries Test | **Priority (Low/Medium/High):** Medium |
| **Test Description:** Simulated ECU receives a request to delete entries in any of the tables | |
| **Acceptance Criteria:** Simulated ECU deletes the request entries from the requested tables | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

## A.5 Security Testing

| | |
|---|---|
| **Test Name:** Database Encryption Test | **Priority (Low/Medium/High):** High |
| **Test Description:** External actor tries to read encryption keys and database on PDM | |
| **Acceptance Criteria:** Data is encrypted and unreadable | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** UWB/CAN Man-in-the-middle Attack Test | **Priority (Low/Medium/High):** High |
| **Test Description:** A malicious actor attempts a man-in-the-middle attack as outlined in section 6.1 | |
| **Acceptance Criteria:** Sensitive packets are symmetrically encrypted | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** UWB/CAN Replay Attack Test | **Priority (Low/Medium/High):** High |
| **Test Description:** A malicious actor attempts a replay attack as outlined in section 6.1 | |
| **Acceptance Criteria:** The RID sends a rolling code, incremented on each packet. The PDM subtracts it's own rolling code value from the decrypted value - if it is negative or larger than a predefined range, the packet is dropped | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** UWB/CAN Scan Attack Test | **Priority (Low/Medium/High):** High |
| **Test Description:** A malicious actor attempts a scan attack as outlined in section 6.1 | |
| **Acceptance Criteria:** The rolling code (the field likely to be attacked) is encrypted by the RID private key and therefor unreadable | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** UWB/CAN Relay Attack Test | **Priority (Low/Medium/High):** High |
| **Test Description:** A malicious actor attempts a relay attack as outlined in section 6.1 | |
| **Acceptance Criteria:** The unlock mechanism will depend on time-of-flight information, therefore the PDM will accurately determine that the RID is too far away and will not unlock | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** UWB/CAN Challenge Forward Prediction Attack Test | **Priority (Low/Medium/High):** High |
| **Test Description:** A malicious actor attempts a challenge forward prediction attack as outlined in section 6.1 | |
| **Acceptance Criteria:** The packet is encrypted and therefor unreadable | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

| | |
|---|---|
| **Test Name:** UWB/CAN Dictionary Attack Test | **Priority (Low/Medium/High):** High |
| **Test Description:** A malicious actor attempts a dictionary attack as outlined in section 6.1 | |
| **Acceptance Criteria:** The packet is encrypted and therefor unreadable, there are no pattern of responses to each packet | |
| **Outcome (Pass/Fail):** | **Date:** |
| **Notes:** | |

# Appendix B: Supporting Design Options

## B.1 Introduction

When considering systems and hardware to implement the PES™, team Echo put careful consideration into what components would best fit the requirements specified in the ongoing requirements document. This appendix explores some of these alternatives and why the specific design selections outlined in this document were made.

## B.2 Wireless Communications

### B.2.1 Technology

In the requirements document, [R-4.1.1.1.a] and [R-4.2.2.1.a] state that the RID and PDM must use transceivers capable of communicating accurate position information with their respective paired devices. This is typically implemented using a *real-time locating system* (RTLS), of which there seem to be three common implementations: Bluetooth with RSSI, Wi-Fi RTLS, and ultra-wideband (UWB). These options are summarized in Table B.2.1.

| Option | Pros | Cons |
|---|---|---|
| Bluetooth | - Lots of support<br>- Inexpensive | - Inaccurate<br>- Unstable |
| Wi-Fi RTLS | - Stable<br>- Precise | - Requires multiple access points<br>- Expensive |
| UWB | - Accurate within 10cm<br>- Inexpensive | - Little support<br>- Unstable if not implemented correctly |

Table B.2.1: Wireless Communication Alternative Design Options

While Bluetooth is typically what comes to mind in local data communications for its compatibility and support, Bluetooth with RSSI implements distance measurements using signal strength, not using time-of-flight. As a result, it does not provide the accuracy that is necessary to perform the motion vector calculations outlined in Section 5.2.1. Wi-Fi RTLS, on the other hand, does use time-of-flight, and could be sufficiently accurate for the PES™. However, because it is still based on Wi-Fi, it requires multiple access points and is too expensive for a small embedded system like the RID.

For these reasons, Bluetooth and Wi-Fi RTLS were eliminated as options for wireless communications by the PES™, and UWB was chosen despite its lack of community support. It's worth noting that Bluetooth

5.1 includes angle of arrival measurements and could potentially work in tandem with UWB to provide even more accurate data, or to replace one of the UWB transceivers. However, due to time constraints and with how new Bluetooth 5.1 is, team Echo decided to stick with one wireless communication protocol for now.

### B.2.2 Hardware

In the PES™ requirements document, the most relevant requirements to UWB hardware are those cited in the previous section. To meet these requirements, the Microchip ATA8350, NXP NCJ29D5, DecaWave DW1000, and DecaWave DW3110 were selected as candidates. The relevant design parameters for each chip are listed in Table B.2.2.

| Option | Parameter | Value |
|---|---|---|
| Microchip ATA8350 [19] | Price | $12.64 |
| | Supply Voltage | 2V - 3.5V |
| | Operating Temperature | -40 °C to 105 °C |
| NXP NCJ29D5 [20] | Price | $17.36 |
| | Supply Voltage | Not found |
| | Operating Temperature | Not found |
| DecaWave DW1000 [21] | Price | $10.76 |
| | Supply Voltage | 2.8V - 3.6V |
| | Operating Temperature | -40 °C -to 85 °C |
| DecaWave DW3110 [22] | Price | $11.41 |
| | Supply Voltage | 2.4V - 3.6V |
| | Operating Temperature | -40 °C to 85 °C |

Table B.2.2: UWB Transceiver Hardware Design Options

The Microchip ATA8350 was not selected because it is quite hard to find, since it's sold out on many websites. Like the NCJ29D5 and DW3110, it's also a newer chip, so the OEM and community support is lacking. While the NCJ29D5 is purchasable, it's quite expensive at $17.36 per chip, and it seems difficult to find any information on its hardware specifications, so it was avoided as well.

The DW1000 transceiver is by far the most popular of the four options, and was the chosen one thanks to its long production run and the variety of suppliers it is available from. In addition to the widely available supply of it, there is a set of examples and a complete user guide including application notes and recommendations. For a technology as new as UWB, this is something that is exceptionally important for mass production and development ease. It is for this reason that the DW3110 was not chosen, as most of the OEM support appears to still be in the beta phase.

## B.3 Interface Technology

In the PES™ requirements document, [R-4.2.2.3.a] states that there must be a bus available for the PDM to interface with that will be used to connect with a manufacturer's motorcycle ECU network. The technology chosen for this purpose was the Controller Area Network (CAN) bus standard. We considered no other options, as CAN is standardized to the point of being ubiquitous in the automotive industry for connecting the on-board ECU systems. To be a viable option for an automotive manufacturer, the PES™ must support CAN.

## B.4 Remote Identifier (RID)

In the PES™ requirements document, [R-4.1.1.3.a] states that the RID must be able to encrypt and decrypt data exchanged over UWB, and [R-4.1.1.1.a] states effectively that the RID must be able to interface with the UWB transceiver. In addition, [R-7.4.1.1.b] states that the battery life of the RID must be at least 12 months. The four microcontroller development boards that were considered to best represent these requirements during prototyping were the STM STM32F103C8T8, STM NUCLEO-L432, STM NUCLEO-F0303, and the Arduino Uno ATmega328P. Detailed information about each of these choices is shown in Table B.4.1.

| Option | Parameter | Value |
|---|---|---|
| STM STM32F103C8T8 development board [23] | Operating Temperature | -40 °C to 85/105/125 °C |
| | Communication | UART, CAN, SPI, I2C, USB |
| | Programming | Serial Wire (SWD) and JTAG programmer |
| | Flash Memory | 64 Kbytes |
| STM NUCLEO-L432 [24] | Operating Temperature | -40 °C - 85/105/125 °C |
| | Communication | UART, CAN, SPI, I2C, USB, SAI |
| | Programming | On-board ST-LINK programmer |
| | Flash Memory | 256 Kbytes |
| STM NUCLEO-F303 [25] | Operating Temperature | -40 °C to 85/105/125 °C |
| | Communication | UART, CAN, SPI, I2C, USB |
| | programming | On-board ST-LINK programmer |
| | Flash Memory | 512 Kbytes |
| Arduino® Uno R3 ATmega328P processor [26] | Operating Temperature | -40 °C to 125 °C |
| | Communication | USART, SPI, I2C |
| | Programming | On-chip debugWire interface |

Table B.4.1: Microcontroller Design Options

The Arduino option was kept in mind as a backup. However, knowing that the PDM will be moved to a custom PCB, the STM32 boards have been prioritized as they work at a lower level, meaning that moving from a development board to a single microcontroller IC will be an easier task. Since the STM32 boards can be paired with the STM32CubeIDE the range of parameters and tools available to developers increase the flexibility in terms of microprocessor usage.

In addition, the DW1000 library for Arduino® cannot take advantage of all the available features from the DWM1000 transceiver, contrary to the C drivers and examples provided by DecaWave specifically for

STM32 microcontrollers, reason why the Arduino® Uno R3 was discarded as an option to implement the PDM.

Along these lines, the STM32F103C8 was eliminated due to the worry it may not meet the processing requirements necessary to perform symmetric encryption and decryption for both CAN and UWB messages. Also, the lack of a built-in ST-LINK meant that one had to be bought separately, increasing the overall costs.

This left the NUCLEO-L432 and NUCLEO-F303. Because the cost of the L432 is significantly lower than the F303 while still meeting the SPI requirements to connect to the UWB transceiver, it was ultimately chosen as the microcontroller for the RID.

## B.5 Proximity Detection Module (PDM)

### B.5.1 Microcontroller

In the PES™ requirements document, [R-4.2.2.1.a] states that the PDM microcontroller must be able to interface with its UWB transceivers and [R-4.2.2.3.a] states that it must be able to interface with its CAN bus. The research for this microcontroller was done at the same time as for the RID's microcontroller, so the detailed information for each option is also listed in Table B.4.1.

The Arduino and the STM32F103C8 were eliminated for the same reasons as listed in the previous section. Between the F303 and L432, the F303 was chosen due to its superior processing power. This decision was made so it could meet the requirements of doing symmetric encryption and decryption for both the CAN and UWB side and also meet the memory requirements to implement the triangulation algorithm for positioning described in Figure 5.2.3.

### B.5.2 CAN Interface

In the PES™ requirements document, [R-4.2.2.3.a] effectively states that the PDM must be designed with hardware that enables its access to a CAN bus. As its microcontroller provides it with access to a CAN controller, a CAN transceiver had to be selected. For this purpose, the Microchip MCP2551, NXP TJA1050, Infineon TLT9251VLE, and Texas Instruments TCAN1051 were selected as potential candidates. The relevant information for these is summarized in Table B.5.1.

| Option | Parameter | Value |
|---|---|---|
| Microchip MCP2551 [27] | Cost | $1.89 |
| | Supply Voltage | 4.5V - 5.5V |
| | Operating Temperature | -40 °C - 85 °C |
| NXP TJA1050 [28] | Cost | $2.73 |
| | Supply Voltage | 4.75 V - 5.25 V |
| | Operating Temperature | -40 °C - 150 °C |
| Infineon TLT9251VLE [29] | Cost | $2.01 |
| | Supply Voltage | 3V - 5.5V |
| | Operating Temperature | -40 °C - 150 °C |
| Texas Instruments TCAN1051 [30] | Cost | $2.15 |
| | Supply Voltage | 4.5 V - 5.5 V |
| | Operating Temperature | -55 °C - 125 °C |

Table B.5.1: PDM CAN Transceiver Design Options

From the microcontroller selected for the PDM, each of the potential options meet an easy-to-supply 5V voltage. They all also work within the temperature ranges that are required for use on a vehicle. For this, it came down to cost and support - the MCP2551 has excellent support while the other options are less commonly used, so getting it working with the CAN bus is the simplest that it could be. Furthermore, the cost for the MCP2551 is the lowest of the options - although it is a small difference, this is the CAN transceiver that will be used during the final production phase. Saving pennies per part could correspond to large savings if the PES™ is manufactured at a larger scale.

## B.6 Simulated ECU

For communication between the simulated ECU and PDM, the Seeed Studio USB to CAN Analyzer Adapter [31] was chosen for its low cost and support provided for development, including drivers for both Windows and Linux and a GUI for Windows to monitor traffic on a CAN bus. Many other CAN-ECU connectors were considered[32] [33] [34] but were not chosen due to their exorbitant (often $200-$800) cost. Other low-cost CAN-ECU hobbyist adapters also exist, like the Canable [35], but are out of stock everywhere or otherwise are no longer being sold.

### B.6.2 Database

There are many database alternatives that could have been used to meet team Echos requirements. The database is required to store various data types (integers, strings, datetimes) and contain unique entries. Most modern database solutions will meet these requirements. However, it is important that team Echo research databases prioritized by popularity. The more popular a database engine, the more likely there is documentation/support available online, as well as a higher likelihood of compatibility with the other technologies used by Echo. Being a small start up company, with no external funding, it is important that team Echo finds a cheap but reliable database engine that fully meets all requirements. Database entries must also be easily transferable between all three subsystems, RID, PDM, and Simulated ECU.

Team Echo chose to use a sqlite database for a few reasons. Firstly, Sqlite is a free database solution with no hosting costs. Although Microsoft SQL Server or MySQL are more “fleshed out”, the cost to hose these databases is too much to justify for the needs of team Echo. A solution like MongoDB would be much more affordable, however, the lack of structure present in a NoSQL database would not be ideal for the future of team Echo data storage. Secondly, Sqlite is one of the most popular modern databases used today [36], allowing for easy documentation and support that will be sourced as needed. Finally, sqlite being stored in a physical file is appealing to team Echo, this will allow for sections of the database to be easily passed over CAN. Potential database options are summarized below in Table B.6.2.

| **Database** | **Specifications** | |
|---|---|---|
| **Microsoft SQL Server** | **Database Structure** | SQL |
| | **Price** | $14000 [37] |
| | **Popularity** | High |
| | **Licensing** | Proprietary |
| **MySQL** | **Database Structure** | SQL |
| | **Price** | $5000 [38] |
| | **Popularity** | Very High |
| | **Licensing** | Open Source |

| | | |
|---|---|---|
| **MongoDB** | **Database Structure** | NoSQL |
| | **Price** | $57/month [39] |
| | **Popularity** | Medium/High |
| | **Licensing** | Open Source |
| **SQLite** | **Database Structure** | SQL |
| | **Price** | Free |
| | **Popularity** | High |
| | **Licensing** | Open Source |

Table B.6.2: Database Design Options

# Appendix C: User Interface and Appearance Design

## C.1 Introduction

The purpose of the PES™ is to provide a fast and accurate keyless system on a motorcycle. Because this product aims to improve a rider's user experience, it is imperative that the PES™ presents an intuitive user interface. Team Echo wants to ensure the PES™ is easy to understand and use for new users.

Within this appendix, the overall user interface design of the PES™ will be discussed. A user analysis, technical analysis and a list of engineering standards applicable to the PES™ product will be examined. The analytical usability testing undertaken by team Echo as well as empirical usability testing to take place with end users will be detailed. Additionally, a graphical representation of the PES™ will be provided to illustrate the user interface design proposed in this section.

## C.2 User Analysis

The user analysis outlines the previous experience of PES™ users and the knowledge/restrictions they are expected to have. The target market for PES™ is electronic motorcycle manufacturers, who would then provide the PES™ system to an experienced and prestigious end user. Being marketed towards both a high-end manufacturing service, and a high-end end user, it is expected that they are both familiar with motorcycles and potentially keyless entry systems.

From the manufacturer's perspective, we want the installation of the PDM system to be seamless. From a design perspective, this involves ensuring the industry standard CAN message protocol will be deployed with our system, allowing for a quick and orderly installation into an existing motorcycle system. The PDM outer casing will contain mounting brackets and installation tools that will be simple to install for anyone with a basic proficiency in motorcycle manufacturing and maintenance. Adding users and RID's to the PES™ database should also be simple for any manufacturing IT team, by using a popular database language (Sqlite) with multi-language support, adding/updating entries will be trivial.

From the end user's perspective, the PES™ system will be no more difficult to use than a standard keyfob already being used by the standard high-end motorcycle driver. Team Echo's keyless entry system will provide a passive entrance system that requires minimal user effort or skill. The back-up button will be large and clearly defined, ensuring no complications to an otherwise keyless system.

## C.3 Technical Analysis

The following section applies the "Seven Elements of UI Interaction" defined by Don Norman to the PES™. These seven elements are: discoverability, feedback, conceptual models, affordances, signifiers, mappings and constraints [40].

### C.3.1 Discoverability

Discoverability provides the user with knowledge of the current state of the system and possible actions. Through the simulated ECU's GUI, the current state of the bike will always be displayed when powered on. The GUI will include three colors to represent the three states of the bike. The brightest color will indicate the current state of the bike. When the system is in the wake-up state, the "Press to start" button will become brighter, indicating that the button can be pressed. Once "Press to start" is selected, the system will move to the start state and the button will become unclickable. Figures C.3.1.1, C.3.1.2, and C.3.1.3 provide mock-up snapshots of the GUI at various states of the system.

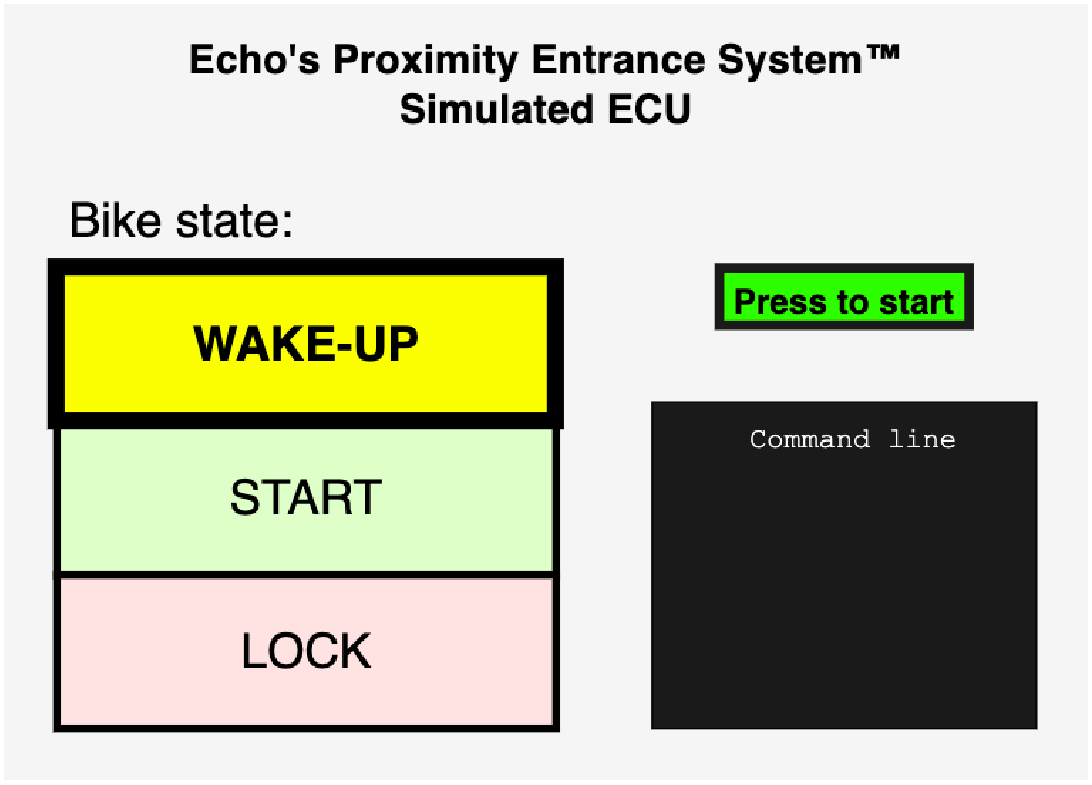


Fig. C.3.1.1: Mock-up of simulated ECU GUI in wake-up state

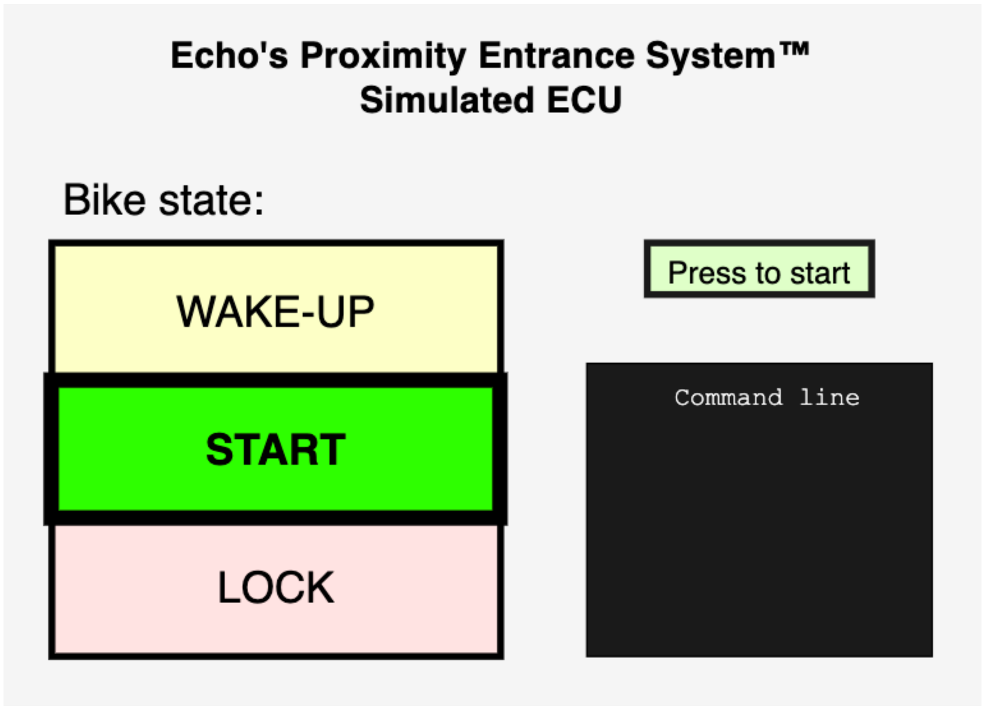


Fig. C.3.1.2: Mock-up of simulated ECU GUI in start state

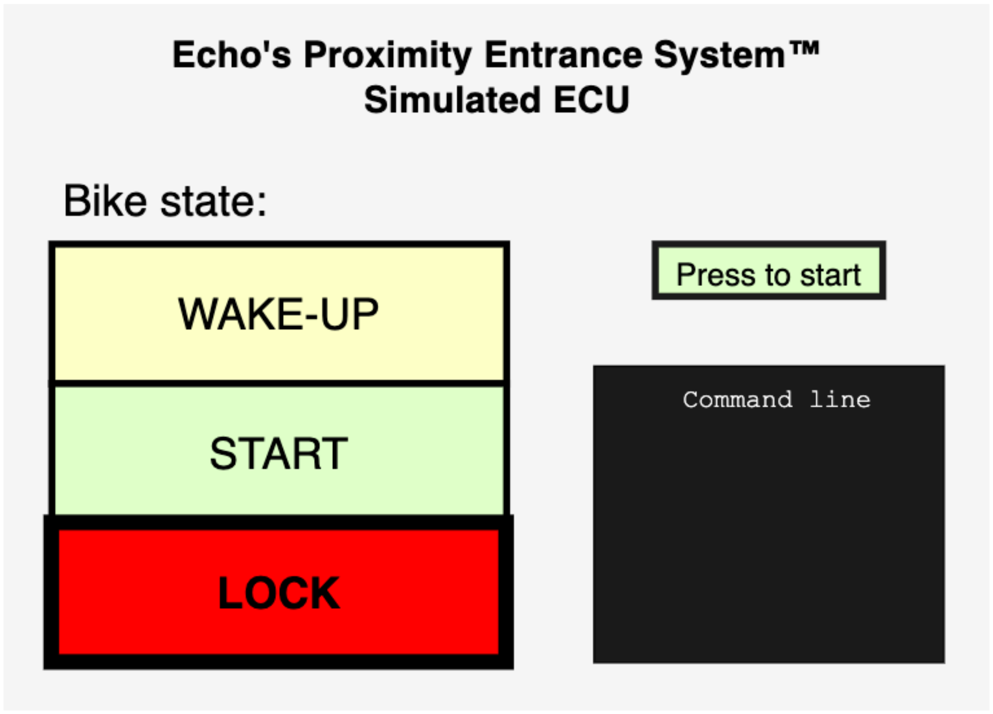


Fig. C.3.1.3: Mock-up of simulated ECU GUI in lock state

## C.3.2 Feedback

Feedback allows a user to receive confirmation that their action has been recognized by the system. As a user interacts with the motorcycle (e.g. walking up to/away from the bike, pressing "start" button), the user will be able to receive immediate feedback from the system by observing the change in bike states. Since the simulated ECU will be used to represent the main ECU of the motorcycle, changes in the bike state will be clearly displayed within the simulated ECU GUI.

## C.3.3 Conceptual models

A conceptual model refers to the user's mental image of how to interact with a system. An ideal design should provide an interface which aligns with the user's expectations. Users of the PES™ will already be familiar with how to ride a motorcycle, so the design of the PES™ will maintain this conceptual model. The driver will not need to change how they approach their motorcycle; they need only to walk up to the bike and drive.

## C.3.4 Affordances

Affordances are the expected action and properties of an object that help determine the operations of the system. The RID is designed with a flat surface to fit comfortably in the pocket of the user. Figure A.3.4.1 illustrates the shape of the device.

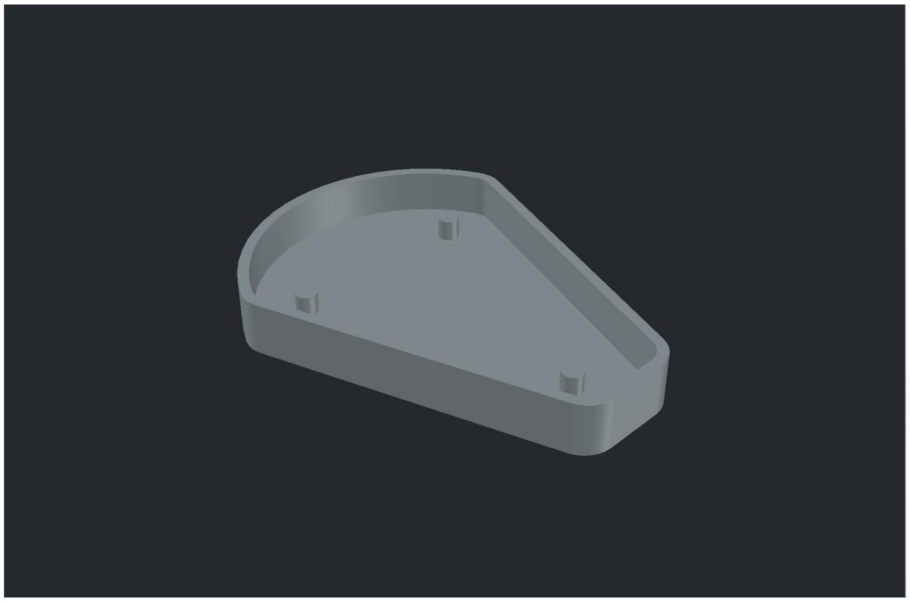


Fig. A.3.4.1: RID Casing mock-up

### C.3.5 Signifiers

Signifiers communicate to the user the behavior of individual components within the system. The RID will emit a chirping sound that is triggered after a successful lock/unlock of the bike. Absence of this chirping sound signifies that the PES™ did not recognize a valid signal.

The simulated ECU GUI will also have a signifier for the user. States of the bike (wakeup, start, lock) will be represented by colors orange, green, and red respectively. This will give a visual representation of the current status of the bike and change as the bike state changes.

### C.3.6 Mappings

Mappings refer to the correlation between the user's movement and the associated action within the system. Due to the design of the PES™, the user goes through a hands-free experience where user controls are not necessary. The PES™ provides signifiers to indicate system feedback to the user.

### C.3.7 Constraints

Constraints provide limitations to the types of actions a user may perform on the system. The user must approach the motorcycle with the proper intent - i.e. the system will not wake up if the user is walking past the bike. The user must press a "start" button located on the handlebars of the motorcycle to start the vehicle. This start button will be included within the simulated ECU's GUI.

## C.4 Engineering Standards

Team Echo will design the PES™ in accordance with the design standards as outlined by IEEE and ISO. The design standards followed are listed in this section.

**ISO 21956:2019** [41]
*Road vehicles — Ergonomics aspects of transport information and control systems — Human machine interface specifications for keyless ignition systems*
Provides human machine interface design specifications for keyless ignition systems that use key code carrying devices.

**ISO 9241-161:2016** [42]
*Ergonomics of human-system interaction — Part 161: Guidance on visual user-interface elements*
Describes visual user-interface elements, requirements, and recommendations on when and how to use them.

**SO/IEC/IEEE 12207:2017** [43]

*Systems and software engineering - Software life cycle processes*

Establishes a common framework for the software life cycle process. Providing terminologies, processes, and activities that are applicable during the development, operation, and maintenance of software systems.

**ISO 11898-1:2015** [44]

*Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling*

Specifies the characteristics of setting up an interchange of digital information between modules implementing the CAN data link layer.

**ISO/IEC 26907:2009** [45]

*Information technology — Telecommunications and information exchange between systems — High-rate ultra-wideband PHY and MAC standard*

Specifies a distributed medium access control sublayer and a physical layer for wireless networks.

**IEEE 802.15.4-2011** [46]

*IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*

Specifies protocol and interconnection for Low-Rate Wireless Personal Area Networks where Ultra-Wideband (UWB) falls.

## C.5 Usability Testing

### C.5.1 Analytical Usability Testing

Team Echo will employ the following usability tests during the development of the PES™. These tests will ensure that end users will be provided with a functional and intuitive system. The goal of analytical usability testing is to find any inaccuracies and errors that may hinder a user's experience, performance, or safety. Therefore, team Echo will base these tests on the following 5 indicators: learnability, efficiency, memorability, errors, and satisfaction as recommended by Michael Sjoerdsma [47].

By following the testing procedure outlined in Figure C.5.1.1 Team Echo will be able to analyze which of the 5 indicators require further enhancements to improve the overall usability of the PES™ system. The analytical usability testing will be performed prior to the empirical usability testing for a couple reasons. Firstly, these tests require no users and can therefore be performed solely by team Echo members. Secondly, these tests will be much easier and faster to perform than the empirical usability tests. However, the downside to the efficiency of these tests is that they are typically less effective at improving the usability of a product.

| **Test** | **Indicator(s)** | **Acceptance criteria** | **Result** |
|---|---|---|---|
| User approaches system the same way they would normally approach their motorcycle | Learnability<br>Efficiency<br>Memorability | Simulated ECU GUI displays bike state changing from lock to wake-up | |
| User selects “start” button located in the simulated ECU GUI after approaching system | Leanability<br>Efficiency | Simulated ECU GUI displays bike state changing from wakeup to start | |
| User walks away from system | Satisfaction | Simulated ECU GUI displays bike state changing from start to lock | |
| Backup button on RID is pressed upon intent detection failure | Errors<br>Satisfaction | Simulated ECU GUI displays bike state changing from lock to wake-up | |
| User enters command through the CLI to modify the database | Efficiency<br>Memorability | Database information is easily modified and results are clearly displayed | |

Table C.5.1: Analytical Usability Testing Procedure

## C.5.2 Empirical Usability Testing

The following section will outline the empirical usability testing that Team Echo will conduct with end users of the PES product. These tests will allow Echo to make any necessary improvements to the PES™

design.The PES™ will be tested by individuals that fit Echo's targeted demographic. These users will have no prior knowledge of the PES™.

A testing session will begin with a meet and greet by a team Echo test monitor, to ensure the user understands their responsibilities and is comfortable. The safety of the user is an important concern for team Echo, as such, the user will then be provided info sheets and instructions on how to interact with the PES™ system safely and correctly. A written consent form will be required before any testing may begin. Before the user begins to interact with the PES™ system they will be given an entry survey as shown in Table C.5.2.1. Once the user has had time to interact with the system, they will be given an exit survey as shown in Table C.5.2.2. While the user is interacting with the system, the test monitor will measure and record the empirical testing metrics outlined in Table C.5.2.3.

| **Question** | **Y/N** | **Additional notes** |
|---|---|---|
| How do you currently start your motorcycle? | | |
| Do you have any experience with other keyless entry systems? | | |
| Where would you prefer to store the RID module? | | |

Table C.5.2.1: Empirical Usability Testing Entry Survey

| **Question** | **Y/N** | **Additional notes** |
|---|---|---|
| Did the PES™ correctly recognize when you were approaching the system for a ride? | | |
| Is there a specific way you approached the bike (intending to ride) where the system did not unlock? | | |
| Is the RID a convenient and portable size? | | |

| | | |
|---|---|---|
| Does the color/label of the back-up button on the RID clearly communicate what its function is? | | |
| Do you prefer the PES™ over other motorcycle entry/exit systems you are familiar with? | | |
| Did the PES™ safely start when the handlebar kill switch is triggered? | | |

Table C.5.2.2: Empirical Usability Testing Exit Survey

| **Variable** | **Result** | **Additional Notes** |
|---|---|---|
| Total Time Performing Testing | | |
| Accuracy | | |
| Errors | | |
| Number of Tasks Completed | | |

Table C.5.2.3: Empirical Usability Testing Test Monitor Record Sheet

## C.7 Conclusion

The goal of the PES™ system is to be as accessible to the Echo target demographic as possible. Because of this, the User Interface and Appearance Design is an integral part of Echo's success. It is important that all user-facing portions of the PES™ system are easy to use for a typical user. Performing a User Analysis from the perspective of the manufacturer and end user while interacting with the PES™ system allowed for a better understanding of the previous experience and knowledge of the intended user base. Following that, a Technical Analysis was employed following Don Norman's "Seven Elements of UI Interaction". Covering topics ranging from, Discoverability and Feedback to Mappings and Constraints ensures that the PES™ UI will be designed intelligently.

By following engineering standards outlined by ISO and IEEE, team Echo is ensuring the PES™ system will remain compliant with industry standards. As the PES™ system is developed, usability testing will be performed both internally (using Analytical Usability testing) as well as externally with potential clients (using Empirical Usability Testing). Team Echo intends to have implemented and tested all UI standards outlined in this document for the proof-of-concept alpha phase prototype. Any future features added in the beta or planned release phase will be tested dynamically during development, following any feedback received from empirical usability testing.