HTWK Hochschule für Technik, Wirtschaft und Kultur Leipzig

October 2024, Leipzig

# Implementation and Evaluation of mobile RSSI-based LoRa Localization

Bachelor Thesis

*Faculty of Computer Science and Media*

*submitted by*

Mose Schmiedel

Prof. Jens Wagner
*First reviewer*

Marian Ulbricht, M.Eng.
*Second reviewer*

*Abstract*

***Keywords:*** *LoRa • RSSI • localization • distance estimation*

# Contents

# 1 Introduction

# 2 State of the Art

LoRa™ and LoRaWAN™ were released as a radio communication technology to the public in 2015 by Semtech. Very early on, it already received coverage in scientific media as it was introduced as a promising long-range radio communication technology for IoT devices [1].

This section presents relevant literature correlating with mobile RSSI-based LoRa™ Localization.

## 2.1 Localization methods

There a multiple approaches for localization in RF based communication networks. The most commonly used all depend on one or more of four physical metrics of the received radio signal. These properties are the angle of arrival (AoA), time of arrival (ToA), Time delay of arrival (TDoA) and received signal strength (RSSI) [2, p. 3].

The presented localization algorithms use fixed reference points with known positions to estimate the position of a device. The reference points will be called anchor nodes and the device will be called end node from now on.

### *2.1.1 Angle of Arrival*

Localization systems which are based on the angle of arrival use a directed antenna array to measure the angle at which the radio signal was received. The AoA information of the signal received at the anchor nodes and the distance between the anchor nodes is used to estimate the position of the end node via triangulation [3, p. 2]. This is done using sine and cosine of the measured angles in combination of the distance between the anchor nodes.

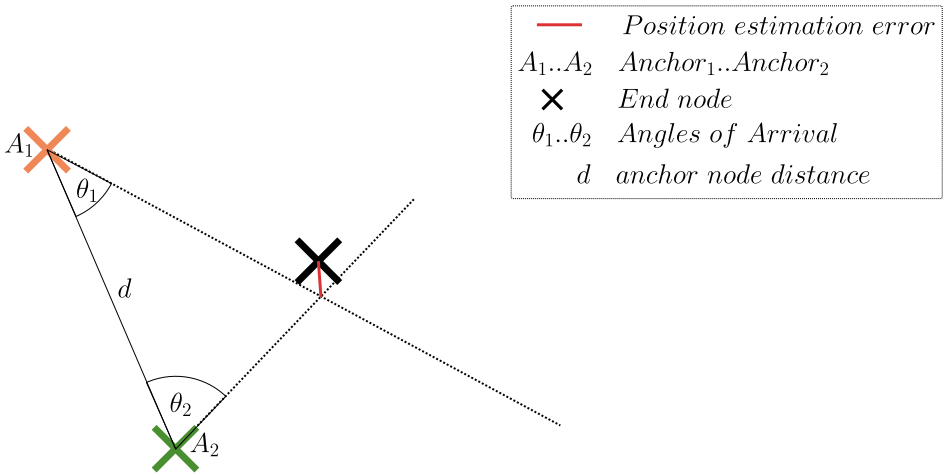


***Figure 1:*** *Triangulation of end node with two anchor nodes*

AoA-based localization can achieve high position accuracy in short range applications. The accuracy is negative correlated with increasing distance between anchor node and end node. This can be simply explained with the following graphic. As can be seen, the resulting change of the position when changing the angle by one degree is greater when the end node is farther away from the anchor node [4, p. 3].

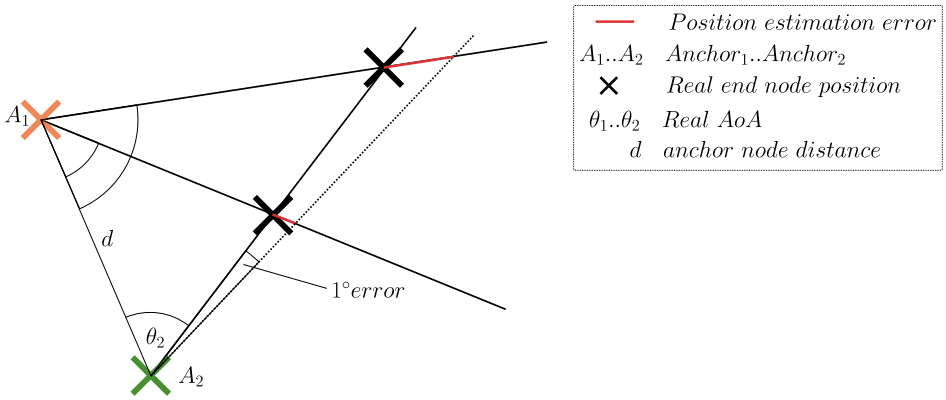


***Figure 2:*** *Triangulation error at different distances*

Another drawback of AoA-based localization methods is that they require additional hardware, not yet commonly implemented in all RF receivers. Also, the need of a directed antenna array makes this localization method unattractive for low-cost applications [4, p. 3].

### *2.1.2 Time of Arrival*

Another localization method presented in the media uses the time of arrival (ToA) of the radio signal. The ToA is used to calculate the time of flight (ToF) which is the time the radio signal needs to travel from the transmitter to the receiver. From this time and the speed of light, the distance between transmitter and receiver can be calculated.

The position of the end node can be estimated when at least three distances between different anchors and the end node can be measured. From these distances, the position can be derived by using trilateration. Trilateration calculates the area of intersection between circles centered at the anchor nodes, each with the measured distance between the corresponding anchor node and the end node. An improved variant of this localization algorithm called multilateration is explained in Section 3.3 in more detail.

ToA-based localization systems need high precision timing capabilities and strict time synchronization between anchor and end node. The following calculation demonstrates that to measure distances up to 1 m the time measurement must provide an accuracy of $\pm\, 3.3\,\mathrm{ns}$.

$$\mathrm{ToF}_{1\mathrm{m}} = \frac{1\,\mathrm{m}}{c} \approx \frac{1\,\mathrm{m}}{3 \cdot 10^8\,\mathrm{m\,s^{-1}}} = 3.3 \cdot 10^{-9}\,\mathrm{s} \triangleq 3.3\,\mathrm{ns} \tag{1}$$

Due to this requirement, ToA-based localization is more suited for long range applications. As with AoA-based localization, this technique also requires specialized hardware not found in every receiver circuit. Clock systems with high resolution and low drift over time and temperature are required to achieve the before mentioned requirements for precision and synchronization of the time measurement.

### *2.1.3 Time Difference of Arrival*

The strict time synchronization of anchor and end node is a requirement which is sometimes very difficult or even impossible to achieve in real world implementations. In such circumstances, it could be required that end nodes can be deployed dynamically such that they are not always powered on and not physically close to each other. These requirements add much complexity to the time synchronization process. One way to deal with this complexity is to avoid it all together by improving the ToA-based

distance measurement. Instead of using the ToF to estimate the distance between anchor and end node, the time difference between the ToA of different anchors is used to calculate the differences of the distances between the anchors and the end node. Through this optimization, time synchronization between anchor and end node is not required anymore, which decreases the complexity of the localization system. The anchor nodes still need to be synchronized so that the differences between the different ToA can be used effectively.

Despite solving the time synchronization challenge of ToA-based localization, TDoA-based localization inherits many of the drawbacks of ToA-based localization. The hardware used for TDoA-based localization still has to provide excellent resolution and drift properties.

#### *2.1.4 Received Signal Strength Indicator*

The last localization method presented in this section uses the Received Signal Strength Indicator (RSSI) of a received radio signal. Like the T(D)oA based method this localization technique also uses trilateration or multilateration to estimate the position of the end node. It differs in the distance estimation technique. Instead of relying on time of flight or a derived measurement, this method uses the decrease in signal strength to estimate the distance between anchor and end node.

The RSSI measured by the receiver depends on many influences like distance between sender and receiver, number of reflections or attenuation by obstacles (buildings, trees, hills) [5, p. 1]. In the literature, several models for radio propagation with focus on different environments or applications are presented. Examples include log-distance model [6, p. 4], Okumura-Hata model [7, p. 4] or Cost 231 model [8, p. 3]. Although there exists no common classification of propagation models, they can be grouped into different categories. One such grouping differentiates the models by the way, they are fitted [5, p. 2].

**Empirical models** rely on intensive measurement of the real behavior of the used radio system.

**Site-specific models** include factors which are derived from a detailed understanding of the environment where the system is later deployed.

**Theoretical models** use the underlying theory of electromagnetic propagation to calculate the path loss in an ideal environment.

In contrast to the other presented localization methods, RSSI-based localization does not need specialized hardware due to RSSI measurement circuit being available in most LoRa™ receiver chips. This makes this method ideal for low-cost applications. This benefit comes with a cost because this localization method cannot achieve accuracies as high as the other presented localization techniques [2, p. 11].

## 2.2 Low Power

LoRa™ is radio communication technology promising low power consumption compared to conventional long range radio communication technologies [1, p. 7]. This is an active field of research because the power budget is an important design factor for mobile applications due to them being limited to battery-based power supplies.

One goal of this thesis is to evaluate RSSI-based localization in a mobile application. This implies, for the above mentioned reasons, that the evaluation of the resulting localization system must include some form of power consumption characterization. To find and evaluate the real power characteristics of devices using LoRa™ communication technology, several studies were performed already. Some of these works are presented in the following section.

Fargas et al. evaluate a LoRa™ localization system in [9] and compare its current consumption with a GPS + GSM based device. They found that their device had significant lower current consumption and therefore could drastically decrease the power requirements for devices with localization systems.

In [10] El Agroudy et al. present an outdoor localization system based on RSSI with ZigBee. They evaluate the power consumption of their implementation and find that it draws around 30 mA in "Active mode" and 6.7 µA in "Sleep mode". "Active mode" is the system state in which communication and localization takes place while "Sleep mode" describes the state in which all system activity is lowered to minimum.

El Agroudy et al. propose, based on the significant difference in current draw between Active and Sleep mode, to reduce the amount of time spent in Active mode by implementing a periodic wake-up event which changes the system state from Sleep mode to Active mode. With this approach the total power consumption of the device can be configured by adjusting the time interval between the wake-up events.

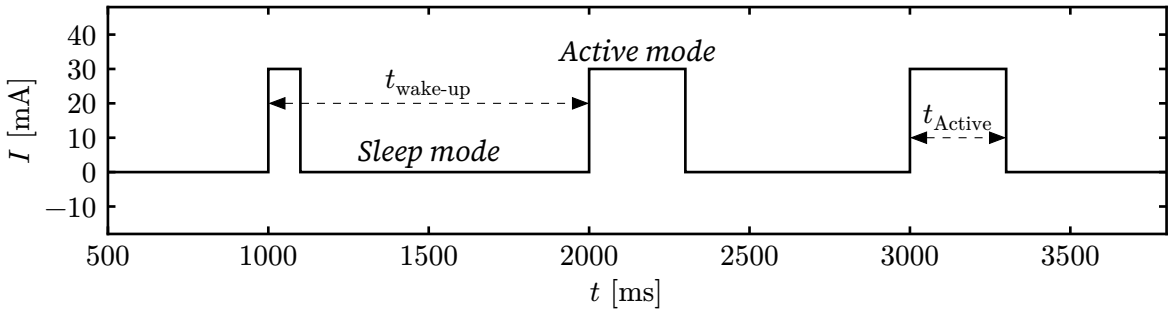


***Figure 3:*** *Current consumption in Active and Sleep mode*

The total current consumption of this approach can be calculated when the current draw and the time spent in the individual modes is known. Following equation shows the final formula. The resulting current consumption is measured in Milliampere hours (mAh).

$$\begin{aligned} t_{\text{wake-up}} &:= \text{Time interval between Wake-Up events} \\ t_{\text{Active}} &:= \text{Time per interval spent in Active mode} \\ I_{\text{total}} &= \frac{I_{\text{Active}} \cdot t_{\text{Active}} + I_{\text{Sleep}} \cdot \left(t_{\text{wake-up}} - t_{\text{Active}}\right)}{t_{\text{wake-up}}} \end{aligned} \tag{2}$$

They used a 620 mAh battery and configured their system to be in the Active mode for 5% of the time. They found that through this method they could improve the life time of their battery from 14.46 hours to 288 hours.

A challenge for this approach is that a device cannot receive any signals while in Sleep mode. This challenge must influence the design of the localization algorithm to ensure that communication between the devices in the localization system is possible. El Agroudy et al. resolve this issue by keeping the anchor node of their localization system always in Active mode so that it can coordinate the communication.

In [11] Mackey et al. evaluate the power consumption of a LoRa™ localization system based on the Dragino LoRa™ v1.3 Shield and an Arduino Uno microcontroller. Like El Agroudy et al. they also implement an localization algorithm working with fixed communication intervals to reduce the amount of time the system is in active mode. They evaluated the power consumption of the receiver node (end node) and the transmitter node (anchor node) separately. They also varied the transmission power and transmission rate and found that they could configure their whole localization system, which consisted of three anchor nodes and one end node, so that it consumed

around 350 mW. They compared this with the power consumption of the GPS module integrated in the Dragino Shield and found that their system outperforms GPS-based localization when at least four end nodes use the deployed anchor nodes.

Despite these promising findings there is still room for improvement. Mackey et al. estimated the life time of a 5000 mAh battery when used as a power source for the anchor or the end node. They found that their anchor node could run up to 200 hours with one battery charge, but their end node would only last for 90 hours. These values would make this solution practical in ad-hoc short-term scenarios where a localization system is need quick and only for a couple of days, but for remote long-term deployments improvements need to be made especially for the power consumption of the anchor node.

Gotthard et al. implement a LoRa™ RSSI-based system in [12], specialized for asset localization in large carparks. They found that with their localization technique a single node of their system could run for about 5 years powered by an 800 mAh CR2 battery. This result looks rather promising compared to the other findings presented in this section. But the system Gotthard et al. builds on fundamental assumptions taken about the environment and scenario the localization is performed in. This includes the assumption that many nodes are deployed in a relatively dense space which simplifies the problem with communication coordination explained earlier.

The assumptions make it difficult to adopt the localization system proposed by Gotthard et al. for general use but they investigate promising ideas which hopefully can be integrated into localization systems in the future to reduce their power consumption.

## 2.3 Similar work

Fargas et al. evaluate LoRa™ for use in an alternative GPS-free geolocation system [9]. Their proposed approach for localization with LoRa™ signals is based on precise measurements of the Time of Arrival (ToA) of one packet at multiple LoRa™ gateways (anchor node). They then calculate the time difference of the different ToA timestamps and estimate the distance between the end node and each gateway by using the propagation velocity of radio waves, which is the speed of light. These distances are then combined by a multilateration algorithm to estimate the position of the end node.

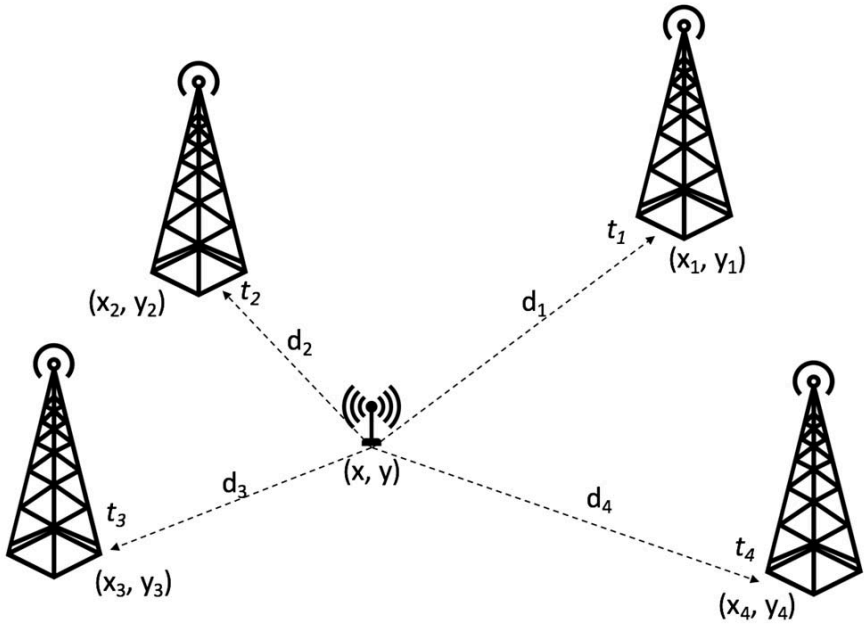


***Figure 4:*** *Position estimation with multilateration algorithm © 2017 IEEE*

Fargas et al. also showed in their work, that by using the Generalized ESD test for detecting and eliminating outliers in the distance measurement data, they could improve the accuracy of the position estimation.

The idea of GPS-less localization is further advanced by Mackey et al. in [11] where they evaluate LoRa™ as alternative localization system for Emergency Services. Instead of using the TDoA method for localizing the end node, they employ RSSI-based localization. For this they use an Arduino Uno microcontroller together with the Dragino LoRa™ v1.3 Shield, which both are readily available off-the-shelf hardware components. They evaluate their implementation on a soccer field of 200 m × 120 m where they achieve positioning accuracies up to 9 m.

In [13] Dieng et al. use RSSI-based LoRa™ Localization to track individual animals of a cattle herd to better deal with livestock theft. Their solution is based on a hybrid localization approach where they use both GPS and LoRa™ RSSI measurements to estimate the positions of the individual animals. They deploy hybrid nodes, nodes equipped with both LoRa™ and GPS, and LoRa-only nodes. The hybrid nodes are used to continuously improve the RSSI-distance model over time by correlating their GPS position with the current distance estimated by the RSSI-distance model.

In [12] Gotthard et al. evaluate RSSI-based LoRa™ localization as asset tracking system for large used car dealerships. They present a novel variant of RSSI-based localization where end nodes only send ping messages between one another. The RSSI acquired from these "pings" are transmitted to a central server, where they can be combined to approximate the position of the individual end nodes. Through this mechanism their proposed system does not require any anchor devices.

## 2.4 Contribution of this Thesis

- implementation and evaluation of mobile localization tag using RSSI-based LoRa™ localization with off-the-shelf (OTS) components
  - RSSI measurement is implemented in nearly all receivers -> no dedicated hardware
  - accuracy loss due to multipath effect should not play a huge role in outdoor localization because of higher LOS (line-of-sight) component of the signal

- evaluation of the feasibility of a LoRa™ based localization system
  - accuracy
  - power consumption
  - frequency band usage

# 3 Principals

This section presents the fundamental theory needed for implementing and evaluating RSSI-based localization systems with LoRa.

## 3.1 LoRa

As already stated LoRa™ is a radio technology proposed by SemTech and developed by the LoRa™ Alliance [1] which primary targets

## 3.2 RSSI-based distance estimation

As already stated in Section 2.1.4 RSSI-based distance estimation

## 3.3 Multilateration

Multilateration is a position estimation algorithm which uses three or more distances between the node which position should be estimated and nodes with known positions. The algorithm can be geometrically explained by drawing circles, each with the measured distance at the anchor as radius, around the positions of the anchors. The position of the end node is estimated by the point of intersection of all the circles. In a perfect scenario this single point of intersection would exist, but in a real-world scenario the distance measurement always includes an error. Due to this error the circles all intersect at different points. These points describe an area in which the real position of the end node must be located.

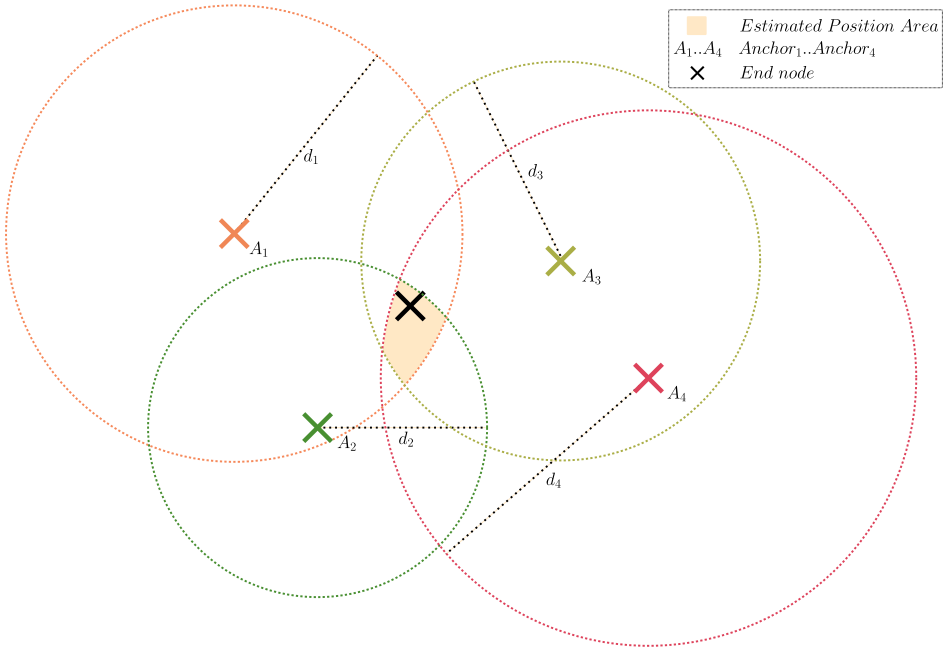


***Figure 5:*** *Position estimation error with multilateration*

## 3.4 Haversine formula

[14]

# 4 Implementation

In the scope of this thesis a simple RSSI-based LoRa™ localization system was implemented. This system and the decisions which led to the final design are presented in this section. For the sake of conciseness from now on the term “localization system” references, where the context does not say otherwise, the localization system that was implemented for this thesis.

## 4.1 System Overview

The localization system implemented for this thesis consists of multiple components. Some of them have physical other have only logical boundaries. The following graphic illustrates the localization system as a whole and gives a quick overview.

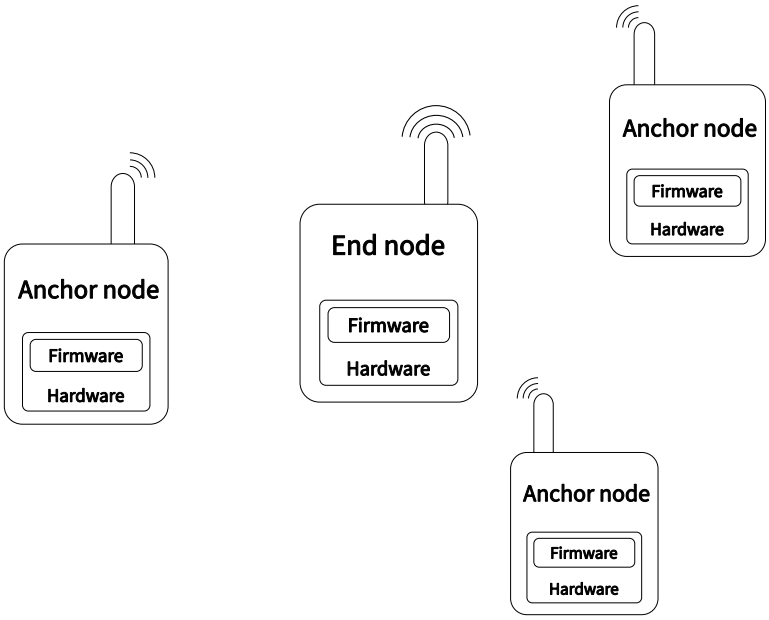


*Figure 6: LoraLocator system overview*

The graphic shows that there are two different types of devices Anchor nodes and End nodes. These serve different purposes in the localization system. The End node is the device which position should be localized. The Anchor nodes serve as fixpoints for the measurements taken to estimate the position of the End node. The system which was implemented and evaluated in this thesis consists of three Anchor nodes and one End node. In-depth explanations of the individual components that make up one Anchor or one End node follow in the subsequent sections of this chapter.

## 4.2 Hardware

Both device types of the localization system are built with the same hardware and differ only in firmware. Every device consist of a NUCLEO-WL55JC [15] this is a evaluation board for the the STM32WL55JC microcontroller [16] developed and manufactured by STMicroelectronics (ST) [17]. This board features LoRa™ transceiver, an integrated programmer and debugger, connectors for the GPIO pins of the microcontroller and easy power supply over Micro-USB. This solution was chosen for several reasons. Most important is of course that the hardware satisfies the basic requirements imposed by the goal of this thesis to implement and evaluate a mobile RSSI-based LoRa™ localization system. This criterium is fulfilled by the NUCLEO-WL55JC evaluation board because it provides an integrated LoRa™ transceiver with built-

in RSSI measurement and promises comparatively low power consumption in the datasheet [18]. Another reason for which the NUCLEO-WL55JC was chosen is the benefit of using prebuilt hardware with a big ecosystem like that of ST. This allows for rapid prototyping by focusing development work on the parts where this localization system differs from previous implementations and reusing boilerplate code where possible. One such part is *SubGHz_Phy* [19, pp. 25-27] which is a driver for the LoRa™ transceiver of the STM32WL55JC.

## 4.3 Firmware

The firmware for the NUCLEO-WL55JC board was written in the programming language C and utilizes prebuilt drivers and libraries provided by STMicroelectronics. The source code was managed with the version control system *git* and uploaded to *GitHub* [20].

The implementation of the firmware is based on the *SubGhz_Phy_PingPong* example application provided by STMicroelectronics [19, pp. 44-46]. The following graphic illustrates the individual logical components the firmware is made of.

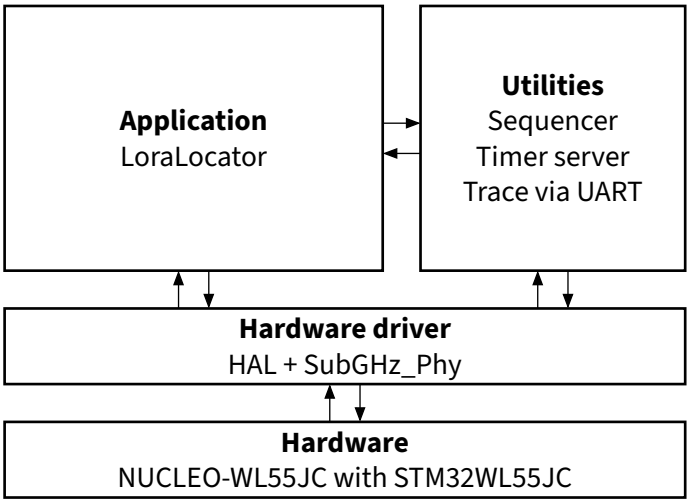


***Figure 7:*** *Firmware architecture*

The main component of the firmware is the *LoraLocator* application. It controls all the other components and implements the logic that is needed for estimating the position of an End node.

### *4.3.1 Conditional compilation*

As described earlier, the firmware of the Anchor node and the End node differ because they both fulfill different tasks in the localization system. Despite there differences these also share some similarities especially in the way the both process incoming and outcoming communication data via LoRa. To avoid unnecessary code duplication both the firmware for the Anchor node and for the End node share the same codebase and node type specific code sections are conditionally included upon compilation depending on which device type was selected via preprocessor macros. For this purpose there are three macro definitions which must be configured when compiling the firmware.

```
// 1=is anchor node; 0=is not anchor node
#define IS_ANCHOR_NODE 1

// 1=is end node; 0=device is not end node
#define IS_END_NODE 0

// unique device identifier
//   - end node: numeric as uint8_t
//   - anchor node: alphabetical as char
#define DEVICE_ID 'A'
```

***Listing 1:*** *Device type and ID configuration*

These three macro definitions are then used inside macro conditionals which control which part of the source code is included in the firmware build. Following code snippet demonstrates a use of a macro conditional.

```
#if IS_ANCHOR_NODE==1 && IS_END_NODE==0
// Anchor node specific code goes here...
#elif IS_ANCHOR_NODE==0 && IS_END_NODE==1
// End node specific code goes here...
#else
#error "Set atleast/only one of IS_ANCHOR_NODE and IS_END_NODE to 1."
#endif
```

***Listing 2:*** *Device type specific code compilation*

#### *4.3.2 Application state machine*

The driving design concept of the LoraLocator application is the finite state machine. The current state is tracked by the `node_state` variable, state changes are performed by the event handlers and the logic executed in a specific state is handled by the core application function `LoraLocator_Process()`.

All application specific state is centralized and managed with a fixed amount of memory. All variables needed can be found at the top of the file `lora_locator_app.c`.

```
typedef enum {
    NODE_STATE_INIT,
    NODE_STATE_INTERVAL_START,
    NODE_STATE_RX_END,
    NODE_STATE_TX_END,
} NodeState_t;

static NodeState_t node_state = NODE_STATE_INIT;
```

***Listing 3:*** *Track last performed action*

The variable `node_state` is used in the core application function `LoraLocator_Process()` to determine the last action that was performed before the core function was called. The possible node states are:

**NODE_STATE_INIT** State the application is in after initialization. Start of the application lifecycle.

**NODE_STATE_INTERVAL_START** State after the main application interval was triggered. `Ping_t` is transmitted or received depending on device type.

**NODE_STATE_RX_END** State after node was receive mode. The received packet is decoded and the next action is triggered depending on device type, received packet type and if reception was successful.

**NODE_STATE_TX_END** State after node was transmission mode. Next action is triggered depending on device type, transmitted packet type and if transmission was successful.

The following listing shows how the behavior of the core function is dependent on the state recorded in the `node_state` variable.

```
void LoraLocator_Process(void) {
    switch(node_state) {
    case NODE_STATE_INIT: {
    // execute NODE_STATE_INIT logic and trigger next action
    }break;
    case NODE_STATE_INTERVAL_START: {
    // execute NODE_STATE_INTERVAL_START logic and trigger next action
    }break;
    case NODE_STATE_RX_END: {
    // execute NODE_STATE_RX_END logic and trigger next action
    }break;
    case NODE_STATE_TX_END: {
    // execute NODE_STATE_TX_END logic and trigger next action
    }break;
    }
}
```

***Listing 4:*** *Execute logic depending on last performed action*

As mentioned before, all state changes occur in event handler functions. These are subroutines that are called when some kind of event occurs, for example when a LoRa™ transmission is successfully terminated. The functions are passed as function pointers to a hardware driver or some other event-based component. The component receiving the event can then delegate the handling of the event to the function defined by the programmer by calling it via the provided function pointer.

```
static RadioEvents_t RadioEvents;
...
RadioEvents.TxDone = &OnTxDone;
RadioEvents.RxDone = &OnRxDone;
RadioEvents.TxTimeout = &OnTxTimeout;
RadioEvents.RxTimeout = &OnRxTimeout;
RadioEvents.RxError = &OnRxError;

Radio.Init(&RadioEvents);
```

***Listing 5:*** *Pass function pointer to radio driver for event handling*

In the next listing the definitions of all event handlers used in the *LoraLocator* app are listed. Note the usage of the prefix `On`, followed by a description of an event, to signal that the function is called when the specified event occurs.

```
// interval_timer elapsed
static void OnIntervalEvent(void *context);

// listen_timer elapsed
static void OnListenEndEvent(void* context);

// Transmission done
static void OnTxDone(void);

// Timeout triggered while transmitting
static void OnTxTimeout(void);

// Timeout triggered while receiving
static void OnRxTimeout(void);

// Error occured during reception
static void OnRxError(void);
```

***Listing 6:*** *List of event handlers used in LoraLocator*

To complete the implementation of the application state machine the state stored in the `node_state` variable must be changed in order to accomplish a transition from one state to another. This is done in the event handler functions by setting the state via the helper function `SetState(NodeState_t next_state)` and calling `QueueLoraLocatorTask()` to signal the scheduler to run the `LoraLocator_Process()` function again. The corresping code can be found in the listing below on lines 5 and 6.

```
static void OnTxDone(void) {
    State = TX;
    switch (tx_result.packet_type) {...} // store and log transmitted packet
    tx_result.state = RESULT_OK;
    SetState(NODE_STATE_TX_END);
    QueueLoraLocatorTask();
}
```

***Listing 7:*** *Example state change in transmission-done event handler*

#### *4.3.3 Hardware drivers*

To develop firmware for the STM32WL55JC microcontroller the hardware drivers developed by STMicroelectronics were used. They provide an abstraction layer over the hardware so that the programmer can access hardware functionality via high level functions instead of configuring the peripheral registers themselves. This abstraction layer has the very creative name HAL, which stands for **H**ardware **A**bstraction **L**ayer. Its documentation can be found here [21].

For controlling the integrated LoRa™ transceiver of the STM32WL55JC microcontroller the *SubGHz_Phy* driver was used additionally. It already implements the most common used functionality like sending and receiving packets via LoRa™ and provides an easy to use API with a single object `Radio` which is used to configure and control the radio transceiver. An example of how the event handlers are configured was demonstrated earlier in Listing 5. Transmission and reception of LoRa™ packets can be triggered by calling methods of the global `Radio` object, demonstrated in the following listing.

```
#define PAYLOAD_LEN 10
uint8_t tx_buffer[PAYLOAD_LEN];
Radio.Send(tx_buffer, PAYLOAD_LEN);

#define RX_TIMEOUT_MS 200
uint8_t rx_buffer[PAYLOAD_LEN];
Radio.Rx(rx_buffer, RX_TIMEOUT_MS);
```

***Listing 8:*** *Switch LoRa™ transceiver to transmission-/reception-mode*

The code above transmits a payload of 10 bytes via a LoRa™ packet, as demonstrated in lines 1 to 3. Lines 4 to 6 show how the transceiver can be configured to receive the transmitted packet. Reception will fail after the time specified by `RX_TIMEOUT_MS` which in this case would be 200 ms. Of course, sending and receiving should be done by separate devices at the same time for them to communicate successfully.

A more detailed explanation of how to build LoRa™ applications with the *SubGHz_Phy* driver and the documentation of the all available methods can be found in here [19].

#### *4.3.4 Utilities*

Several utility modules distributed by STMicroelectronics were used to simplify the firmware development process. They helped reduce the amount of boilerplate code, i.e. source code required for basic project setup that has little variation between different projects. Figure 7 lists all the utility modules used for the final firmware. This section briefly introduces the two most influential modules.

The firmware uses the **Sequencer** module to schedule the execution of the `LoraLocator_Process()` function. This allows the event handler functions, which are often executed in an interrupt context, to indirectly call the core function and leave the interrupt context quickly. The documentation of the Sequencer module can be found here [22].

```
UTIL_SEQ_RegTask(
    (1 << CFG_SEQ_Task_SubGHz_Phy_App_Process),
    UTIL_SEQ_RFU,
    LoraLocator_Process);
```

***Listing 9:*** *Sequencer task registration*

```
static void QueueLoraLocatorTask() {
    UTIL_SEQ_SetTask(
        (1 << CFG_SEQ_Task_SubGHz_Phy_App_Process),
        CFG_SEQ_Prio_0);
}
```

***Listing 10:*** *Helper function to schedule task execution*

Basic timing functionality is implemented in the *LoraLocator* application with the **Timer server** module. This module allows to create an arbitrary amount of independent timers. A timer generates a timeout event and calls a provided event handler function after a fixed amount of time provided upon creation. The time is specified in Milliseconds. Unfortunately no official documentation could be found for this module but most important functionality can be understood by directly reading the source code.

```
/* Timer that triggers `LoraLocator_Process` periodically to either transmit a
`Ping_t` (end node) or listen for a `Ping_t` (anchor node). */
static UTIL_TIMER_Object_t interval_timer;
...
UTIL_TIMER_Status_t timer_result = UTIL_TIMER_Create(
        &interval_timer,      // timer object
        INTERVAL_PERIOD_MS,   // timeout value
        UTIL_TIMER_PERIODIC,  // timer mode (ONESHOT=run once and stop,
                              // PERIODIC=run and restart until stopped)
        &OnIntervalEvent,     // callback function to call when timer elapses
        NULL);                // argument passed to callback function

if (timer_result != UTIL_TIMER_OK) {
    // handle timer creation error
    APP_LOG(TS_ON, VLEVEL_M, "Could not create interval timer.\n\r");
}
```

***Listing 11:*** *Timer creation*

```
UTIL_TIMER_StartWithPeriod(&interval_timer, INTERVAL_PERIOD_MS);
```

***Listing 12:*** *Start timer with specified timeout value*

## 4.4 Distance estimation

- End node sends ping

```
typedef struct {
    uint8_t device_id;
    uint8_t packet_id;
} EndNodeRequest_t;
```

- Anchor node responds with measured RSSI

```
typedef struct {
    Device_t anchor_id;
    int16_t recv_rssi;
} AnchorResponse_t;
```

- experiment in park

## 4.5 Localization

- ACK of distance measurement ping

# 5 Evaluation

## 5.1 Distance estimation

- log-distance model must be fitted
  - RSSI measured at different distances
  - multiple measurements per distance (ca. 80) → calculate average RSSI
  -

| Nr | Description | avg. Error |
|---|---|---|
| #01-1 | Beschte | 2 m |
| #01-2 | Beschte | 2 m |
| #02-1 | Beschte | 2 m |
| #02-2 | Beschte | 2 m |

***Table 1:*** *Experiments for distance estimation evaluation*

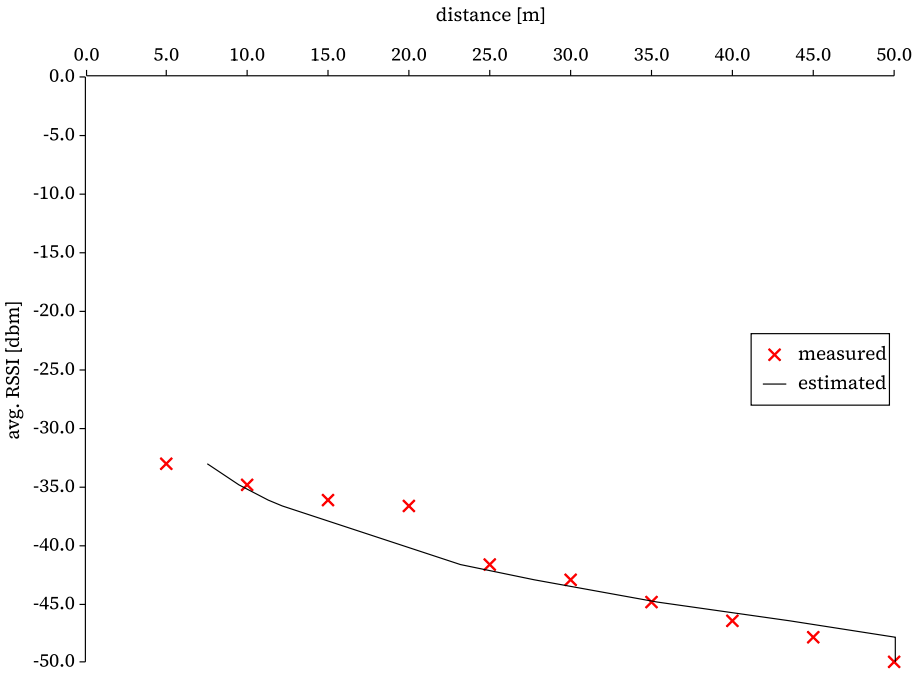


***Figure 8:*** *Experiment #02-2 distance estimation*

## 5.2 Localization

- define Anchor A as (0,0)
- measure distances between Anchors and calculates cartesian coordinates with haversine formula

# 6 Future Works

# List of Figures

## List of Tables

## List of Listings

# Bibliography


[1] L. Vangelista, A. Zanella, and M. Zorzi, "Long-Range IoT Technologies: The Dawn of LoRa™," in *Future Access Enablers for Ubiquitous and Intelligent Infrastructures*, V. Atanasovski and A. Leon-Garcia, Eds., Cham: Springer International Publishing, 2015, pp. 51–58. doi: 10.1007/978-3-319-27072-2_7.

[2] L. E. Marquez and M. Calle, "Understanding LoRa-Based Localization: Foundations and Challenges," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11185–11198, Jul. 2023, doi: 10.1109/JIOT.2023.3248860.

[3] R. Peng and M. L. Sichitiu, "Angle of Arrival Localization for Wireless Sensor Networks," in *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, Sep. 2006, pp. 374–382. doi: 10.1109/SAHCN.2006.288442.

[4] M. Zare, R. Battulwar, J. Seamons, and J. Sattarvand, "Applications of Wireless Indoor Positioning Systems and Technologies in Underground Mining: a Review," *Mining, Metallurgy & Exploration*, vol. 38, no. 6, pp. 2307–2322, Dec. 2021, doi: 10.1007/s42461-021-00476-x.

[5] J. A. Azevedo and F. Mendonça, "A Critical Review of the Propagation Models Employed in LoRa Systems," *Sensors*, vol. 24, no. 12, p. 3877–3878, Jan. 2024, doi: 10.3390/s24123877.

[6] G. M. Bianco, R. Giuliano, G. Marrocco, F. Mazzenga, and A. Mejia-Aguilar, "LoRa System for Search and Rescue: Path-Loss Models and Procedures in Mountain Scenarios," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1985–1999, Feb. 2021, doi: 10.1109/JIOT.2020.3017044.

[7] A. I. Griva *et al.*, "LoRa-Based IoT Network Assessment in Rural and Urban Scenarios," *Sensors*, vol. 23, no. 3, p. 1695–1696, Jan. 2023, doi: 10.3390/s23031695.

[8] M. Stusek *et al.*, "Accuracy Assessment and Cross-Validation of LPWAN Propagation Models in Urban Scenarios," *IEEE Access*, vol. 8, pp. 154625–154636, 2020, doi: 10.1109/ACCESS.2020.3016042.

[9] B. C. Fargas and M. N. Petersen, "GPS-free geolocation using LoRa in low-power WANs," in *2017 Global Internet of Things Summit (GIoTS)*, Jun. 2017, pp. 1–6. doi: 10.1109/GIOTS.2017.8016251.

[10] N. El Agroudy, N. Joram, and F. Ellinger, "Low power RSSI outdoor localization system," in *2016 12th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, Jun. 2016, pp. 1–4. doi: 10.1109/PRIME.2016.7519456.

[11] A. Mackey and P. Spachos, "LoRa-based Localization System for Emergency Services in GPS-less Environments," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Paris, France: IEEE, Apr. 2019, pp. 939–944. doi: 10.1109/INFCOMW.2019.8845189.

[12] P. Gotthard and T. Jankech, "Low-Cost Car Park Localization Using RSSI in Supervised LoRa Mesh Networks," in *2018 15th Workshop on Positioning, Navigation and Communications (WPNC)*, Oct. 2018, pp. 1–6. doi: 10.1109/WPNC.2018.8555792.

[13] O. Dieng, C. Pham, and O. Thiare, "Outdoor Localization and Distance Estimation Based on Dynamic RSSI Measurements in LoRa Networks: Application to Cattle Rustling Prevention," in *2019 International Conference on Wireless and Mo-*

*bile Computing, Networking and Communications (WiMob)*, Barcelona, Spain: IEEE, Oct. 2019, pp. 1–6. doi: 10.1109/WiMOB.2019.8923542.

[14] B. Gardiner, W. Ahmad, T. Cooper, M. Haveard, J. Holt, and S. Biaz, "Collision Avoidance Techniques for Unmanned Aerial Vehicles Technical Report # CSSE 1101," 2011. Accessed: Sep. 30, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Collision-Avoidance-Techniques-for-Unmanned-Aerial-Gardiner/49d4db3e71beb1795838c0e06fa4a0335a3608d5

[15] "NUCLEO-WL55JC - STM32 Nucleo-64 development board with STM32WL55JC MCU, SMPS, supports Arduino and morpho connectivity - STMicroelectronics." Accessed: Oct. 14, 2024. [Online]. Available: https://www.st.com/en/evaluation-tools/nucleo-wl55jc.html

[16] "STM32WL55JC - Sub-GHz Wireless Microcontrollers. Dual-core Arm Cortex-M4/M0+ @48 MHz with 256 Kbytes of Flash memory, 64 Kbytes of SRAM. LoRa, (G)FSK, (G)MSK, BPSK modulations. AES 256-bit. Multiprotocol System-on-Chip. - STMicroelectronics." Accessed: Oct. 14, 2024. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32wl55jc.html

[17] "STMicroelectronics: Our technology starts with you." Accessed: Oct. 14, 2024. [Online]. Available: https://www.st.com/content/st_com/en.html

[18] "STM32WL55xx STM32WL54xx - Multiprotocol LPWAN dual core 32-bit Arm® Cortex ® -M4/M0+ LoRa® , (G)FSK, (G)MSK, BPSK, up to 256KB flash, 64KB SRAM." Accessed: Oct. 14, 2024. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32wl55jc.pdf

[19] "AN5406: How to build a LoRa® application with STM32CubeWL." Accessed: Oct. 14, 2024. [Online]. Available: https://www.st.com/resource/en/application_note/an5406-how-to-build-a-lora-application-with-stm32cubewl-stmicroelectronics.pdf

[20] M. Schmiedel, "moseschmiedel/lora-locator." Accessed: Oct. 14, 2024. [Online]. Available: https://github.com/moseschmiedel/lora-locator

[21] "UM2642: Description of STM32WL HAL and low-layer drivers." Accessed: Oct. 15, 2024. [Online]. Available: https://www.st.com/content/ccc/resource/technical/document/user_manual/group1/6f/be/85/55/8c/26/4c/22/DM00660673/files/DM00660673.pdf/jcr:content/translations/en.DM00660673.pdf

[22] "Utility:Sequencer - stm32mcu." Accessed: Oct. 15, 2024. [Online]. Available: https://wiki.st.com/stm32mcu/wiki/Utility:Sequencer