Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

DCF77 Based Long-Term Timer

Semester Thesis

Jianwei Sun

jiansun@student.ethz.ch

Computer Engineering and Networks Laboratory
Department of Information Technology and Electrical Engineering
ETH Zürich

Supervisors:
Roman Trüb
Prof. Dr. Lothar Thiele

January 29, 2018

Acknowledgements

This semester thesis would not be what it is without the tremendous support of Roman Trüb, who not only supervised all aspects of this investigation from research to development to testing, but also provided helpful insight and encouraging suggestions to guide this project to completion. Thank you.

Abstract

Wireless IoT devices are increasing in popularity due to their versatility in numerous applications, such as in sensing in remote environments. However, as their popularity increases, so does their complexity, resulting in tighter requirements in energy consumption without sacrificing performance. Communication between these devices contributes significantly to energy consumption, so efficient channel utilization protocols are necessary. However, such schemes often require that the devices' internal clocks be synchronized, and is usually achieved by sending additional data packets. As a result, other synchronization sources, such as the DCF77 longwave time radio signal, are investigated for their feasibility in replacing traditional synchronization schemes to reduce channel utilization. Compared with receivers for other accurate time sources, such as GPS, DCF77 receivers have lower power requirements, which makes the usage of DCF77 ideal for power-sensitive wireless IoT devices. This semester thesis explores the advantages and limitation of DCF77 in terms of timekeeping accuracy, reliability, and power consumption; and, proposes a synchronization scheme based on a low-cost and low-power off-the-shelf DCF77 receiver.

Table of Contents

List of Symbols

α	Proportional of time spent in LPM3.5 relative to total time
$e(T)$	Relative frequency error as a function of temperature in Celcius
F_t	Timeseries frequency
K_d	PID differentiating constant
K_p	PID proportional constant
P_{ACTIVE}	Power consumption of timer device in active mode
$P_{LPM3.5}$	Power consumption of timer device in LPM3.5 mode
P_{TOTAL}	Total power consumption of timer device
T	Variable representing temperature in degrees Celcius
T_d	Minimum amount of time that could result in a clock drift of T_p
$T_{LPM3.5}$	Amount of time spent in LPM3.5 mode
T_p	Duration of time in which a full DCF77 frame can be received with probability, p
T_{TARGET}	Target time of end event relative to start event
T_t	Timeseries environmental temperature
ACTIVE	Union of the AWAKE and LPM3 power states
ADC	Analog-to-digital converter
AM	Amplitude modulation
AWAKE	MSP430 power state with CPU and peripherals powered
CEST	Central European Summer Time
CPU	Central processing unit inside the MSP430
DCF77	Longwave time signal / transmitter
DMA	MSP430 Direct Memory Access controller
GPS	Global position system
LPM3	MSP430 power state with CPU powered off
LPM3.5	MSP430 power state with everything except for the RTC powered off
MSP430	Texas Instrument's line of low-power microcontrollers
PIC	A family of microcontrollers from Microchip
PM	Phase modulation
RTC	MSP430 Real Time Clock hardware module
ublox	GPS receiver, from company u-blox, that provides accurate reference waveforms

Chapter 1

Introduction

1.1 Motivation

Wireless IoT devices are increasing in popularity due to their versatility in numerous applications, such sensing in remote environments. Being battery powered, these devices must operate conservatively since wireless transmitting and receiving have significant energy requirements. A possible strategy to conserve battery energy is to limit sending and receiving. For example, a transmitting and receiving pair of devices would agree upon a future time to communicate again; and until then, both devices would operate in a low power sleep state. However, such a strategy is prone to clock drift errors - if one device wakes up too early or too late, then the transmitted data packet could be lost. Therefore, the need for time synchronization is an additional requirement of these devices. Typically, additional data packets that encode time information are exchanged so that all receivers can synchronize themselves to the transmitting device. However, since channel capacity is a scarce resource that also relates to power consumption, it would be ideal to achieve synchronization by other means. This need motivated the consideration of radio time signals as a possible synchronization source. Specifically, the DCF77 signal is promising for a variety of reasons.

Traditionally used as a synchronization source for consumer radio clocks and watches, the DCF77 longwave time signal is broadcast in Mainflingen, Germany, and covers a large part of continental Europe [5]. Its carrier frequency of 77.5 kHz is not only highly accurate as it is generated by an atomic clock, but is in a frequency band typically not used by IoT devices [5]. The longwave nature of the signal makes it easily pass through building obstacles, which increases its effectiveness for indoor devices. Furthermore, the signal is able to be received and decoded by low-cost and low-power AM receivers, which is an advantage over GPS-based time signal receivers, as the latter is more complex and has significantly higher energy requirements. Motivated by these benefits, this semester thesis aims to investigate how DCF77 can be used as an effective source for clock synchronization in IoT devices.

1.2 Goals

This project first aims to quantify the quality of DCF77 (accuracy and availability) in order to determine any limitations of the signal. Then, this project aims to develop a low-power hardware prototype timer device running a synchronization algorithm that is able to generate an event at a preset time in the future, while ensuring its internal clock's drift is adequately compensated by synchronizing to DCF77. Lastly, the project aims to evaluate the time keeping accuracy of the timer device relative to a highly accurate reference GPS time source, and then characterize the timer device's power consumption for comparison to other DCF77-based synchronization devices in literature.

1.3 Outline

This semester thesis is organized as follows: Chapter 2 Background provides necessary information about the DCF77 signal as well as mentions related works in synchronization using DCF77. Chapter 3 Methodology focuses on the tools used for this project, the development approach of the timer device, and the outline for the structure of the project. Chapter 4 Preliminary Studies discusses the clock drift and DCF77 experiments and provides analysis of the results. Chapter 5 Timer Device then details the conceptual design, implementation and performance evaluation of the timer device. Lastly, Chapter 6 concludes with final remarks.

Chapter 2

Background

2.1 DCF77 Signal

The DCF77 time signal is a longwave radio signal broadcast from Mainflingen, Germany by Physikalisch-Technische Bundesanstalt (PTB)[5]. The main purpose of this broadcast is to disseminate accurate CEST time across most of continental Europe for radio-based clocks and devices to synchronize [5]. The signal's high accuracy is attributed to the atomic clock based generation of the 77.5 KHz carrier frequency, which is estimated to deviate from its nominal value by less than a relative amount of $\pm 2 * 10^{-13}$ [5]. The DCF77 radio wave consists of a ground wave, which follows the curvature of the Earth, and a sky wave, which gets reflected by Earth's ionosphere [2]. The reflection allows the sky wave to travel farther distances. There are three distinct regions around the DCF77 source: within 600 km in which the ground wave dominates; between 600 km and 1100 km in which the ground and sky waves constructively and destructively interfere with each other; and beyond 1100 km in which the sky waves dominate [2].

CEST time is encoded into the DCF77 signal through means of amplitude modulation (AM) and phase modulation (PM) schemes [5]. In the AM scheme, each frame of data lasts for one minute and consists of 59 data bits that encode the CEST time, parity bits, leap year bits, civil warnings, and weather broadcast messages as shown in Fig. 2.1 [5]. The bits are represented in the AM scheme by reducing the carrier wave amplitude to 15% of its nominal value for 100 milliseconds and 200 milliseconds to represent digital bits 0 and 1, respectively.

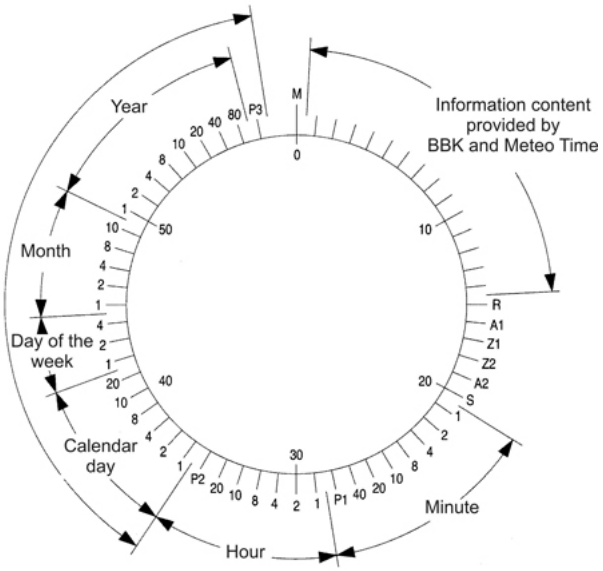


Figure 2.1: Encoded data scheme in each frame of DCF77 transmission [5]

The AM time code can be extracted by using an envelope detector circuit with a hardware counter, so hobbyist DCF77 receivers can be made with very low cost. The PM schemes encode the same information, but require more sophisticated hardware to receiver. However, higher accuracy can be expected from using the PM scheme [5].

2.2 Related Work

Existing literature has demonstrated the possibilities of using DCF77 as an AM synchronization source can provide accuracy on the order of milliseconds [2]. Although not as accurate as GPS, DCF77 based receivers can outperform GPS receivers in power consumption by more than a factor of 1000 [2]. In [2], Chen et al. develop a low-cost prototype that is capable of receiving and decoding the DCF77 signal, and demonstrates its ability in transmitting accurately decoded CEST timestamps as well as its power consumption in sleep and awake modes [2]. The study also provides some measurements on cumulative clock drift over an extended time period and measurements of DCF77 availability by counting the number of entire received frames over a few days. Chen et al.'s study most strongly advocates the low power capabilities of their timer prototype. They demonstrate the possibility of microwatt-levels of power consumption, which is orders of magnitude lower than GPS-based devices, by using a low-cost off-the-shelf DCF77 receiver with a very low-power PIC microcontroller. By characterizing the power consumption over microcontroller active and sleep modes, Chen

et al. also provide insight into what factors contribute most significantly to the overall power consumption. In this case, the power consumed during sleep states between DCF77's one second pulses contributes a significant portion to the overall averaged power consumption [2]. Their observation has motivated this semester thesis' choice of microcontroller, which will be discussed in a later section. Despite providing great detail on power consumption analysis and measurement statistics, Chen et al. only briefly mention that their prototype timer device is capable of achieving millisecond accuracy by comparing the decoded CEST time with a more accurate reference, and that its pulse-per-second output is generated by the PIC microcontroller's internal clock, whose clock drift is corrected by receiving a couple of pulse-per-second DCF77 pulses [2]. However, the device's time code output requires a full frame of DCF77 to be decoded in order to update its internal count of CEST time. As shown in the same study, the ability to decode full minute frames of DCF77 varies throughout the day. As a result, the device's accuracy will be sub optimal if its clock drifts and synchronization cannot occur before it has to generate timing events for other devices. This limitation will be explored further in this semester thesis by considering the possibility of synchronization without using entire frames.

A different study conducted by Laveyne et al. in [3] contrasts Chen et al.'s study by only considering the start of the power reduction in DCF77's signal amplitude, which signifies the transmission of a single bit of data, rather than using an entire frame. The method is much simpler due to less computation overhead required for getting a timestamp and is less susceptible to DCF77 unavailability: single bits of data can be received with higher probability than an entire frame consisting of 60 consecutive bits. In Laveyne et al.'s study, different sensors try to receive the same start of power reduction event in order to synchronize among themselves [3]. However, Laveyne et al. then demonstrates that sensor placement affects the relative delay of the received event between the sensors, by up to 0.67 milliseconds on average [3]. The accuracy is substantially lower compared to that of GPS-based synchronization receivers; however, Laveyne et al. state that the accuracy is sufficient for low cost applications [3]. Laveyne et al.'s study has informed that different sensors placed in far-spaced locations will receive DCF77 sourced events at different times, which limits how far sensors can be placed, and the types of obstacles around them, to ensure a certain level of accuracy. However, the different sensors are still able to measure the same elapsed duration of time to high accuracy, assuming that the delay from the DCF77 power reduction event to each timer's reception of that event is constant for each device. Hence, if the sensors were originally synchronized, they will be able to remain synchronized if they each measured elapsed time between their received DCF77 sourced events, rather than relying on all sensors to receive subsequent events at the exact same time.

Chapter 3

Methodology

In this chapter, ways of realizing the timer device is discussed by considering necessary hardware required for implementation and testing. Furthermore, the approach taken to develop the timer device from concept to implementation to evaluation will also be explored. This chapter starts off by discussing the selection process for the microcontroller platform and the type of DCF77 receiver used in the timer device. Then, the chapter explores the GPS time reference used in project experiments as well as archived DCF77 data used to compare test results. Finally, the chapter presents the approach taken to build and evaluate the timer device.

3.1 Platform

3.1.1 Microcontroller Selection

Different families of low power microcontrollers were considered for this semester project. The most important criteria for selection were power consumption and ease of implementation, given the short time frame of the project. The candidate microcontroller families selected for final consideration were the Atmel SAML21 [16], Microchip's PIC16/18LF series [17][18], the NXP KL02 entry level microcontroller [19], ST's STM32L021 [20], and Texas Instrument's MSP430FR low power series [9]. These choices of microcontrollers were motivated both by literature, such as using the same PIC microcontroller from [2], and prior experience working with the microcontroller. Power consumption data was gathered from the microcontrollers' datasheets and tabulated in Table. 3.1 for comparison.

Note that not all data was available across all datasheets. For example, the SAML21 did not provide power consumption data for 1 MHz clock at $3.3V$, which made direct comparison more difficult. The MSP430FR5969 microcontroller was chosen as the most appropriate microcontroller to use due to availability and its ease of development with its Launchpad Development Board [9].

Microcontroller	**Supply** (V)	**Clock (MHz)**	**Awake** (μW)	**Sleep** (μW)
SAML21	1.8	4	410 - 1051	9.4 - 19.4
	3.3	4	554 - 1386	14.5 - 30.4
PIC16LF	1.8	1	198 - 360	0.036 - 1.8
	3.0	1	594 - 1188	0.09 - 3.27
PIC18LF	1.8	1	306 - 396	0.036 - 1.8
	3.0	1	832 - 1099	0.24 - 6.6
KL02	3.0	4	495 - 651	0.99 - 1.2
STM32L021	1.2	1	168 - 216	44 - 204
MSP430FR5969	3.0	1	630	1.8 - 2.7

Table 3.1: Comparison of candidate microcontrollers

3.1.2 DCF77 Receiver Hardware

Since DCF77 is commonly used by radio clocks as a source of synchronization, off-the-shelf hardware for receiving or decoding DCF77 is easily available. The simplest receivers quantize the DCF77 signal so that a microcontroller can interpret it. More complicated receivers are able to decode the quantized signals and extract the encoded CEST time. Moreover, some receivers demodulate the AM component of the DCF77 signal, whereas more complicated receivers use PM for improved accuracy [5].

Since the wavelength of DCF77 is too long, electrical antennas of a reasonable length cannot be used. As a result, magnetic antennas consisting of a copper coil wound around a ferrite core are commonly used. This choice of antenna imposes orientation requirements to ensure that the magnetic fields of the DCF77 signal oscillate through the copper coil, resulting in an induced electrical current that can be filtered and demodulated.

In the interest of power savings, the simpler receivers that only quantized the DCF77 signal were explored in this semester project. A sample of an off-the-shelf receiver ordered from Conrad is shown in Fig. 3.1. The receiver includes a $40mm$ ferrite antenna that is soldered onto the PCB.

The Conrad receiver, similar to many other off-the-shelf receivers, provides a simple open-drain output for interfacing to microcontrollers. Both polarities of the quantized signal are available for increased versatility. In general, the DCF77 receivers are simple and straightforward to use.

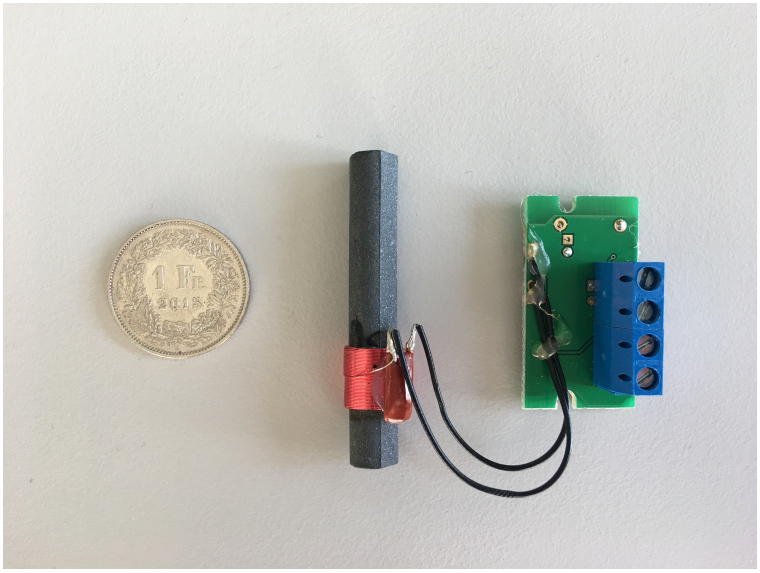


Figure 3.1: Sample DCF77 receiver ordered from Conrad

3.1.3 RocketLogger Measurement Device

The RocketLogger is a high resolution voltage and current measurement device capable of automatically switching between different ranges of measurements [12]. This functionality was necessary for measuring transitions between the different sleep and awake power states of the timer device. The RocketLogger also provides a web-based GUI to easily plot and log the measured current and voltage data. The ease of use and automatic range switching capabilities of the RocketLogger motivated its use in this semester thesis.

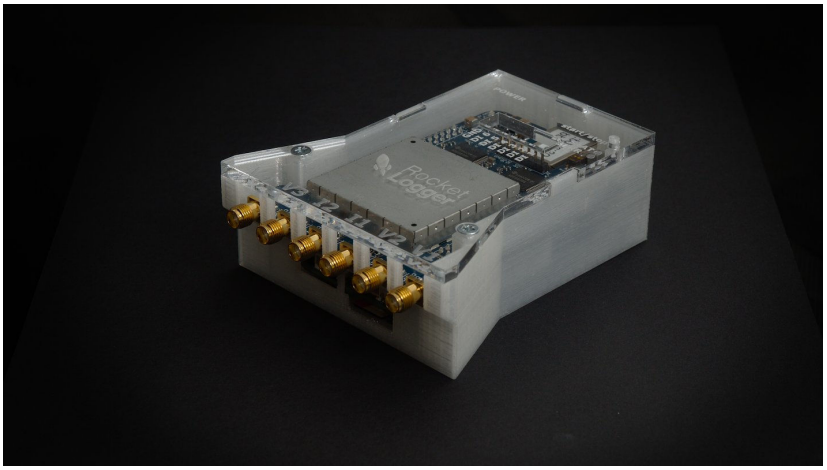


Figure 3.2: RocketLogger voltage and current measurement device [12]

3.2 GPS as a Ground Truth

In order to perform accurate time measurements of a timer device, another time source with higher accuracy than the timer device was needed. To satisfy this requirement, GPS was received with the ublox LEA-6T GPS receiver, shown in Fig. 3.3. The receiver is able to output a reference periodic waveform, called a timepulse, with frequency ranging from 1 Hz to 10 MHz. The timepulse is synchronized to GPS to provide a highly accurate and precise frequency reference that does not drift substantially over time [11]. Datasheet values report timing accuracy of less than $60ns$ [11]. The timepulse can be used as a reference for measuring timed events, or as a clock source for a microcontroller test bench. Relying on the accuracy and precision of the GPS-sourced timepulse meant that all measurements made were as good as that of the ublox GPS receiver. Even though GPS still has inaccuracies, it is as close as possible to a perfect source that we could practically and easily use. As a result, the GPS was taken as the ground truth for all timing measurements.

Figure 3.3: ublox evaluation module

3.3 Reference DCF77 Data

Archived DCF77 data available from [14] was used as a reference for the analysis in the Preliminary Studies section of this semester report. The reference data is collected by dcf77logs.de in real time, archived, and made available for download. The archived data consists of decoded bitstreams organized by frames and translated into a human-readable date format. A sample of the data is shown in Fig. 3.4.

```
0 12345678901234 567890 12345678 9012345 678901 234 56789 012345678
M Wetterdaten    Info   Minute P StundeP Tag    WoT Monat Jahr    P  Datum:       Zeit:
=============================================================================================
0 11011001111000 001001 00000000 0000000 001000 101 00100 000100001  Fr, 04.04.08 00:00:00, SZ
0 01100010100111 001001 10000001 0000000 001000 101 00100 000100001  Fr, 04.04.08 00:01:00, SZ
0 00001010110011 001001 01000001 0000000 001000 101 00100 000100001  Fr, 04.04.08 00:02:00, SZ
0 01111000101100 001001 11000000 0000000 001000 101 00100 000100001  Fr, 04.04.08 00:03:00, SZ
0 00110000011101 001001 00100001 0000000 001000 101 00100 000100001  Fr, 04.04.08 00:04:00, SZ
0 01101101001111 001001 10100000 0000000 001000 101 00100 000100001  Fr, 04.04.08 00:05:00, SZ
0 10010111000011 001001 01100000 0000000 001000 101 00100 000100001  Fr, 04.04.08 00:06:00, SZ
0 01010100100001 001001 11100001 0000000 001000 101 00100 000100001  Fr, 04.04.08 00:07:00, SZ
0 01111111001101 001001 00010001 0000000 001000 101 00100 000100001  Fr, 04.04.08 00:08:00, SZ
0 00100011000101 001001 10010000 0000000 001000 101 00100 000100001  Fr, 04.04.08 00:09:00, SZ
```

Figure 3.4: Sample of DCF77 transmission collected by dcf77logs.de [14]

The reference data was useful in determining which bits should have been received during analysis. By comparing each received bit to the corresponding archived bit, the analysis could determine exactly which pulses in the waveform of the DCF77 output were noise and which ones corresponded to the DCF77 signal. Since dcf77logs.de is not officially recognized by PTB, the correctness of all recorded DCF77 data from this source is not guaranteed. However, all of the decoded data used in analysis showed correct parity checks and decoded to CEST timestamps that agreed in relation to adjacent timestamps. As a result, it was assumed that the archived logs from dcf77logs.de were reliable for the purpose of being reference data.

3.4 Approach

The complete organization of this semester thesis can be summarized by the following plan of record in Fig. 3.5. Each of the blocks in the diagram represents a separate standalone task, and the arrows represent the dependencies between the tasks. Each of these blocks is described further under the subsections of this chapter.

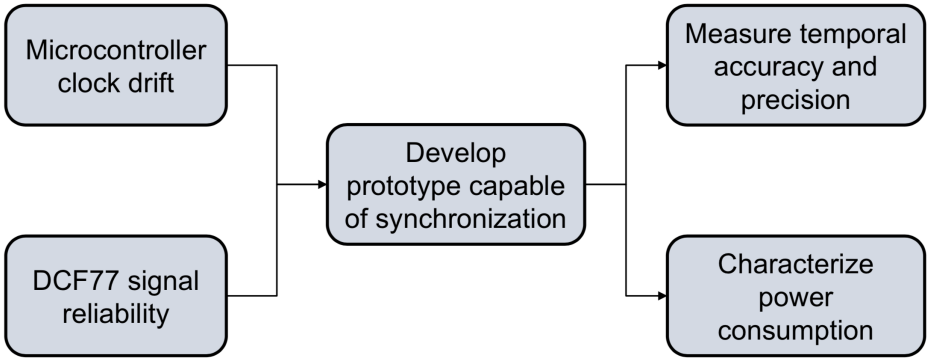


Figure 3.5: Task dependencies of this semester thesis [4]

3.4.1 Preliminary Studies

In order to develop a timer device that synchronizes to DCF77, it was first necessary to understand the problem that had motivated the need for synchronization devices. Specifically, understanding how clock drift is affected by environmental temperature can provide quantitative requirements for synchronization; such as, how often synchronization must occur to satisfy a target level of accuracy. These results could be determined by studying the change of clock frequency as a function of environmental temperature. Such a model could then be helpful in the development of the timer device's synchronization algorithm and be useful in providing an upper bound on the rate of clock drift that the timer device has to correct.

In addition to understanding the problem of clock drift, it was important to study the DCF77 signal by looking at its reliability, accuracy, and quality. Determining the limitations of using DCF77 as a synchronization source guided the development of the timer device's synchronization algorithm to be robust to those limitations. An experiment specifically measuring the quality of DCF77 could log timestamps of each received bit in the broadcast over a long duration. By analyzing the received bits and checking for extra or missing ones through comparison with the archived DCF77 data, it was possible to determine the quality of DCF77 reception over time, the timestamp accuracy of each received bit, as well as overall robustness to environmental interference. The conclusions from this experiment would not only provide insight into the benefits of using DCF77 as a synchronization source, but also suggest what limitations a DCF77 based synchronization algorithm must overcome.

3.4.2 Timer Device

The timer device, whose block diagram is shown in Fig. 3.6, is a hardware device that can generate timed events for preset durations. For example, when the timer device receives a start event, it can wait for a preset duration of time and then generate an end event. Start and end events could be rising edges on GPIO pins or even more complicated procedures; the exact event can be application specific. Analogous to an alarm clock, the timer device is able to count time until it is required to trigger some end event.

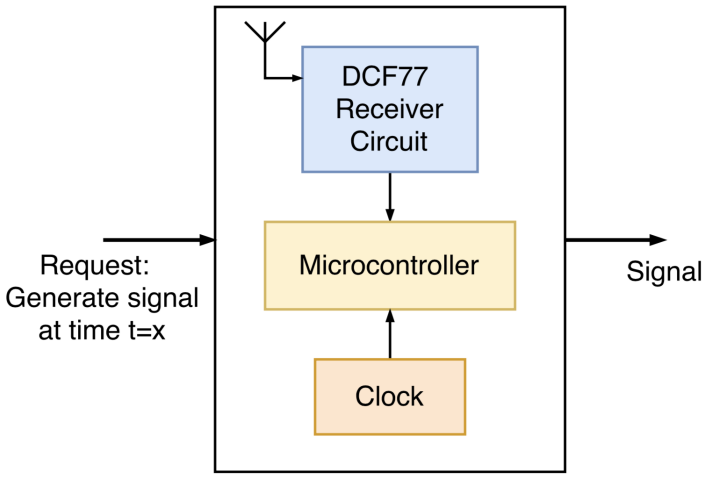


Figure 3.6: Block diagram of the timer device [1]

The timer device needed to able to count time for up to 24 hours according to project requirements [1]. To do so, the timer device also included a DCF77 receiver, which it used as a synchronization source.

The development of the timer device was influenced by the results from the Preliminary Studies on clock drift and DCF77. The Clock Drift study provided a bound on how much clock drift the timer device had to compensate. The DCF77 Accuracy and Availability study indicated the quality of the synchronization source that the timer device used, and showed any limitations that had to be considered by the timer device.

3.4.3 Evaluation

Accuracy

To measure the accuracy of the timer device in a controlled way, a test bench was used to programmatically send start events to the timer device and measured the elapsed time until the timer device generated the end events. The measured duration was then compared to expected values to determine how much the timer device was being affected by clock drift, and how well it compensated.

Power Consumption

To satisfy power requirements of the timer device, it was necessary to fully characterize all current consumptions of the components in the device during the different modes of operation. Having a complete set of power consumption data could then provide meaningful results, such as peak and average power consumption. These results were useful in comparing the performance of the timer device to the requirements of similar devices in literature.

Chapter 4

Preliminary Studies

4.1 Temperature Dependency of Clock Drift

4.1.1 Concept

In typical microcontroller applications, the processor is clocked by a signal generated from either internal oscillators or external resonators or crystals [10]. Electrical or piezoelectric clock sources exist with trade-offs between absolute frequency accuracy, Allan deviation, power consumption, and material cost. As no clock source is perfect, a clock source will exhibit undesired frequency dependency on external factors; the largest of which is temperature [2][15]. Since it is known that temperature impacts clock accuracy, oscillator datasheets usually provide upper bounds on the frequency change per change in unit temperature. For a sense of scale, the internal oscillator of the MSP430FR5969 used as a test platform in this semester thesis reports typical datasheet values of $0.01\%/^{\circ}\mathrm{C}$ at a supply voltage of $3.0V$ [9].

With regards to the goal of developing a timer device that is resilient to clock drifts caused by temperature, it was helpful to understand exactly how clock drift is affected by temperature. Once this relationship was determined, maximum bounds on amount of clock drift per unit time could be calculated. These bounds helped to determine how long a timer device could go without synchronization in order to still be accurate to a desired amount.

4.1.2 Experiment Setup

The following experiment was motivated by the need to understand the relationship between frequency of a typical oscillator and the environmental temperature. In this experiment, the frequency of the internal oscillator and the environmental temperatures were recorded periodically. By recording the temperature and the clock frequency as timeseries data, a model could be derived between the two data sets. To precisely measure the internal oscillator's frequency, the ublox pulse-per-second output was used as a reference connected to an interrupt-enabled input pin on the MSP430. At the GPS-synchronized rate of 1 Hz, provided by the ublox receiver, the MSP430 was interrupted to sample its clock frequency and temperature. Its frequency was sampled by measuring the clock cycles elapsed between each pulse-per-second rising edge. The temperature data was directly read from the internal temperature sensor of the microcontroller using an internal 12-bit ADC. The valid range of the temperature sensor measured with 12-bit resolution resulted in a granularity of approximately 0.115°C per increment. The results were then stored in the internal non-volatile memory of the MSP to be read after the experiment had completed. A block diagram of the experimental setup is shown in Fig. 4.1 and a photograph of the setup is shown in Fig. 4.2. To vary the environmental temperature over a long period of time, the experiment setup was placed in a weather-proof box outdoors for 46 hours. By relying on the temperature change between daytime and nighttime, the experiment captured a realistic range of temperatures that a typical microcontroller in an outdoor setting would be exposed to.

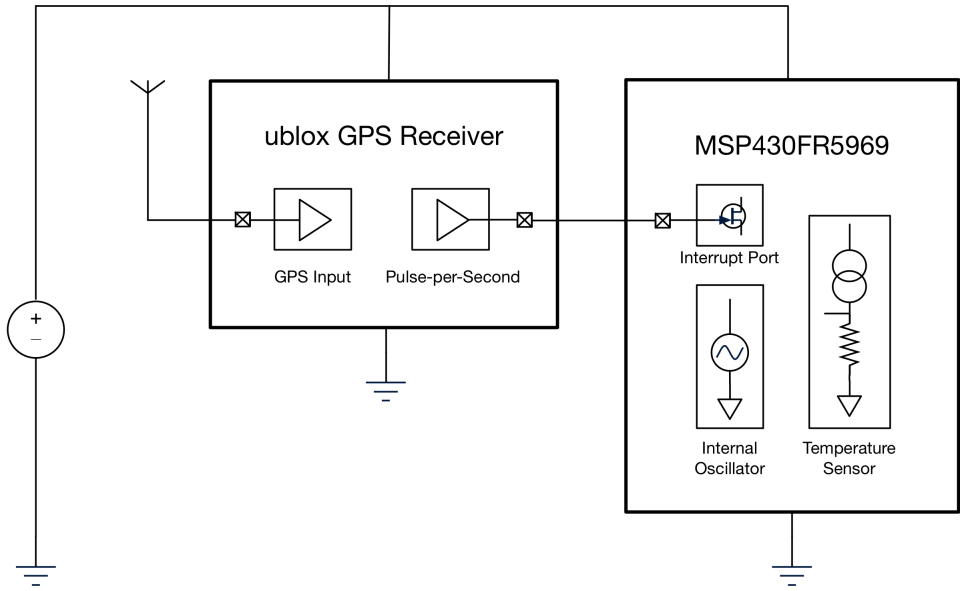


Figure 4.1: Hardware setup showing the MSP with ublox GPS module

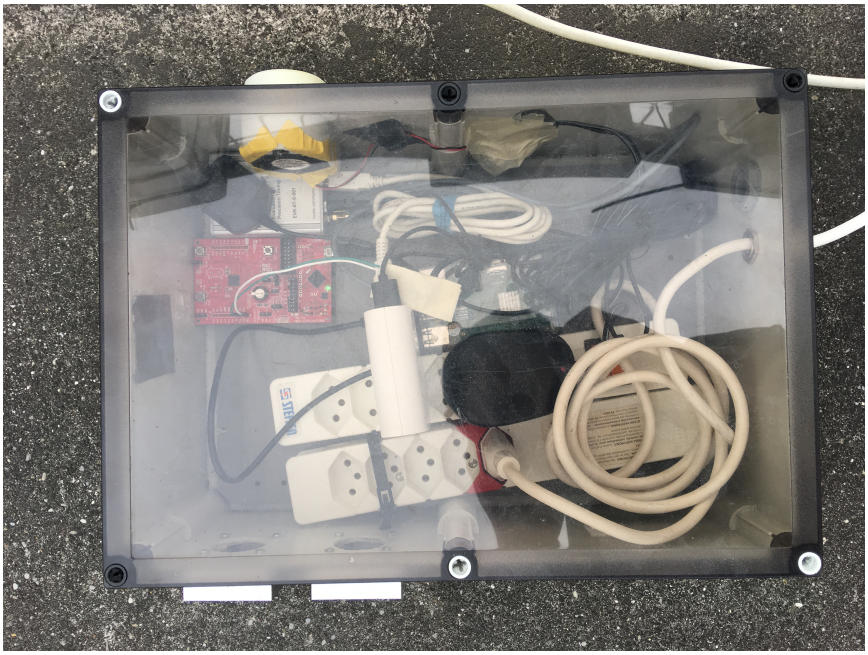


Figure 4.2: Photograph of the weather-proof box with test setup inside on 30 November, 2017 at ETH Zürich

4.1.3 Results

The experimental data consisted of temperature and frequency timeseries data that was used to fit a frequency-temperature model. High frequency noise in the temperature timeseries data was filtered since temperature was not expected to change rapidly. In post-processing, the temperature was passed through a non-causal low pass filter to remove noise above 0.001 Hz. The following results were observed.

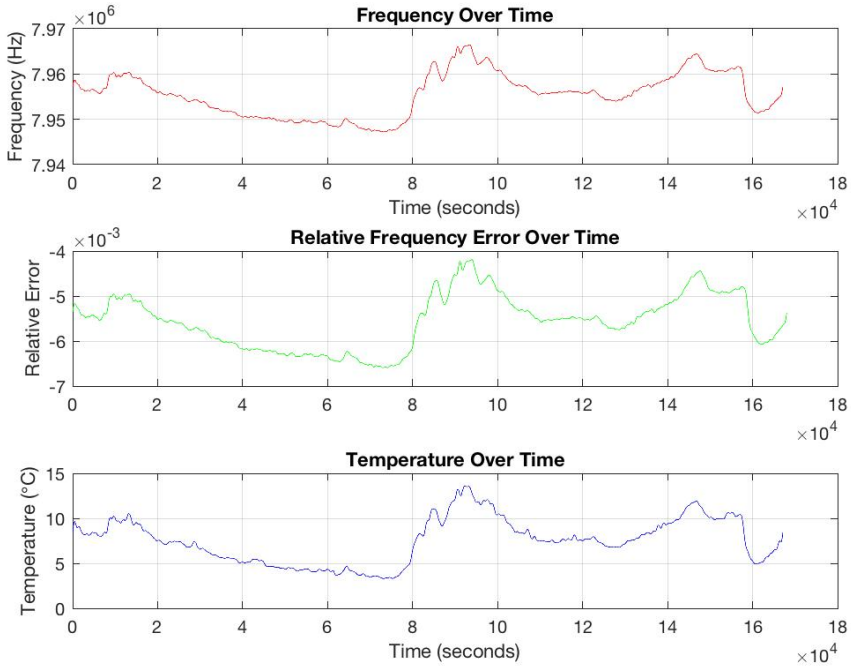


Figure 4.3: Plot of clock frequency, relative frequency error, and temperature over testing time

4.1.4 Analysis and Discussion

Initial qualitative observations were that there is a direct relationship between the environmental temperature and the frequency deviation from nominal value of the internal oscillator. Since the rate of change of temperature was mild, time lags between changes in temperature and corresponding changes in clock frequency were not easily observed. For these reasons, a simple first order model of the form $e(T) = a * T + b$ was fit, where $e(t)$ is the relative frequency error as a function of temperature, T, given in degrees Celcius. The model parameters,

a, b, were determined by solving a Least-Squares minimization, whose solution is the Moore-Penrose pseudo-inverse.

$$e(T) = 0.00023565 * T - 0.00733542 \tag{4.1}$$

By plotting the relative frequency error directly against temperature, the accuracy of the model fit was observed in 4.4. The slope of the fitted line, a, is the change in frequency error per change in temperature.

$$\frac{de}{dT} = a \approx 0.00024 \tag{4.2}$$

For a 20°C change in temperature, a drift from the nominal value of 8 MHz by as much as 0.5% could be expected, which is approximately one in every two hundred seconds.

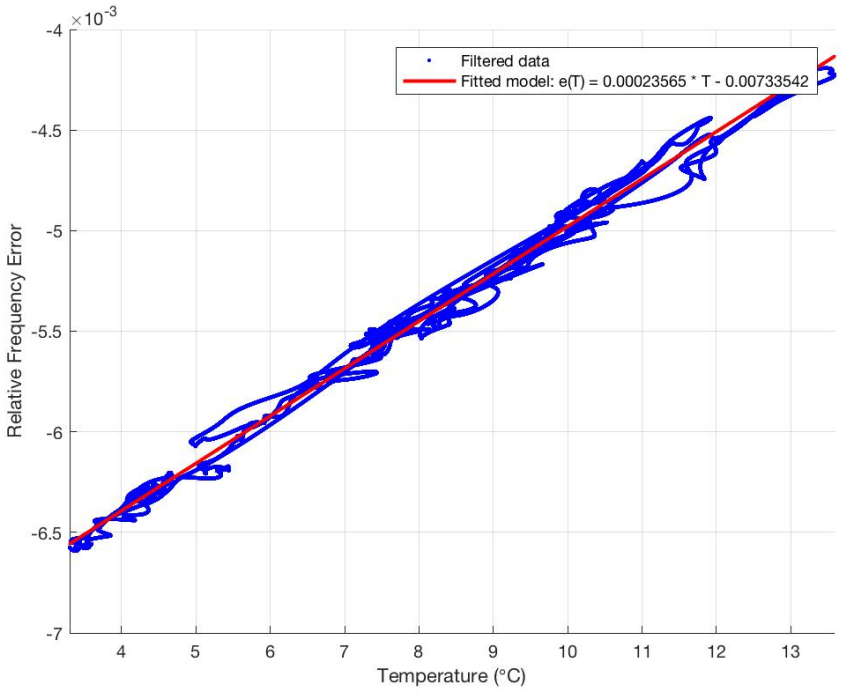


Figure 4.4: Plot of relative frequency error as a function of temperature

The relationship was modeled using a first order equation for simplicity, but for temperatures outside the studied range, this is usually not the case [15].

However, for the purpose of studying temperature changes in realistic outdoor conditions, the first order model was sufficient. The fitted data suggested that increases in temperature resulted in increases in frequency relative to the nominal value. The higher frequencies would cause the internal timer of a microcontroller to increment at an increased rate, which would make the timer generate events sooner. The error of generating events too early (or too late if the frequency is below the nominal value) is exacerbated by longer durations, which allows for the clock to drift even more. The results of this experiment in the context of this semester thesis showed that without synchronization, microcontrollers are prone to clock drifts, especially if subject to even mildly varying temperatures. Even at room temperatures, a long enough duration could result in significant clock drift errors.

4.2 Reliability Study of DCF77 Receivers

4.2.1 Concept

The DCF77 signal can be received by simple low cost AM receivers that are readily available off-the-shelf. According to [13], interference from nearby LCD screens, signal blockage due to walls, and noise from the mains power lines affect the quality of the DCF77 receivers. To better understand the magnitude of each source of interference, a qualitative study was performed. In the study, the DCF77 receiver under test were exposed to each type of interference separately and the quantized DCF77 signal output was observed for erroneous behavior. The outline of this experiment is tabulated in Table. 4.1.

Test Number	Description
1	Interference from LCD screens
2	Signal blockage from concrete walls
3	Wall socket noise

Table 4.1: Qualitative Tests for DCF77 Interference

4.2.2 Hardware and Implementation

The receiver under test, shown in Fig. 4.5 was powered by a battery source and had its quantized DCF77 output connected to an LED. This setup easily allowed visual observations to be made in order to qualitatively determine if the interference source affected the quality of the receiver. The setup was placed in each of the interference sources mentioned in Table. 4.1 and the output on the LED was compared to the expected behavior, which was produced in the absence of interference.

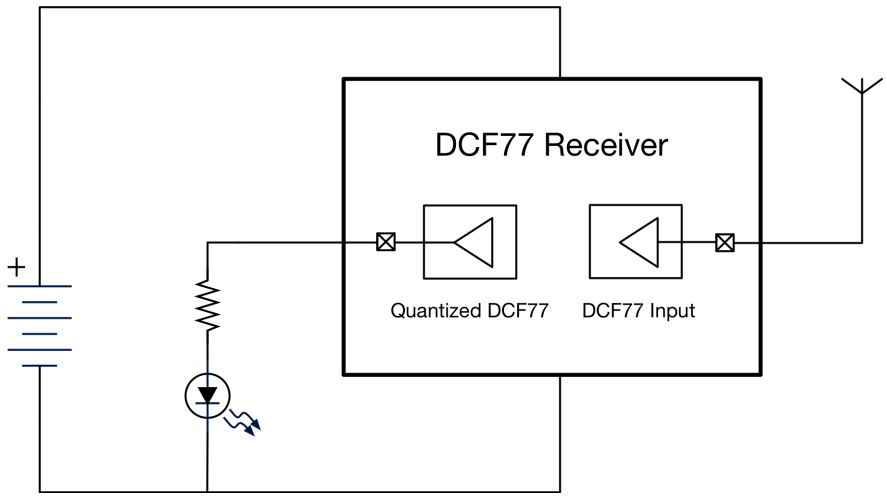


Figure 4.5: Block diagram of LED connected to DCF77 receiver

In this experiment, three commonly available DCF77 receivers were investigated. The models tested were ordered from online distributor Conrad, online store Reichelt Electronik, and HKW Electronik GmbH. The Conrad receiver shown in Fig. 4.6 was designed around the T4224 time code receiver [6]. The Reichelt receiver shown in Fig. 4.7 most likely used the MAS1017 time code receiver according to its online photo, but the shipped product did not have any part number markings. Lastly, the HKW receiver (output polarity "n") shown in Fig. 4.8 also did not list any part numbers of the time code receiver used. Unlike the first two receivers, the HKW receiver allowed for different lengthed antennas to be used, so three different lengths (40mm, 60mm, 100mm), shown in Fig. 4.9, were ordered as well.

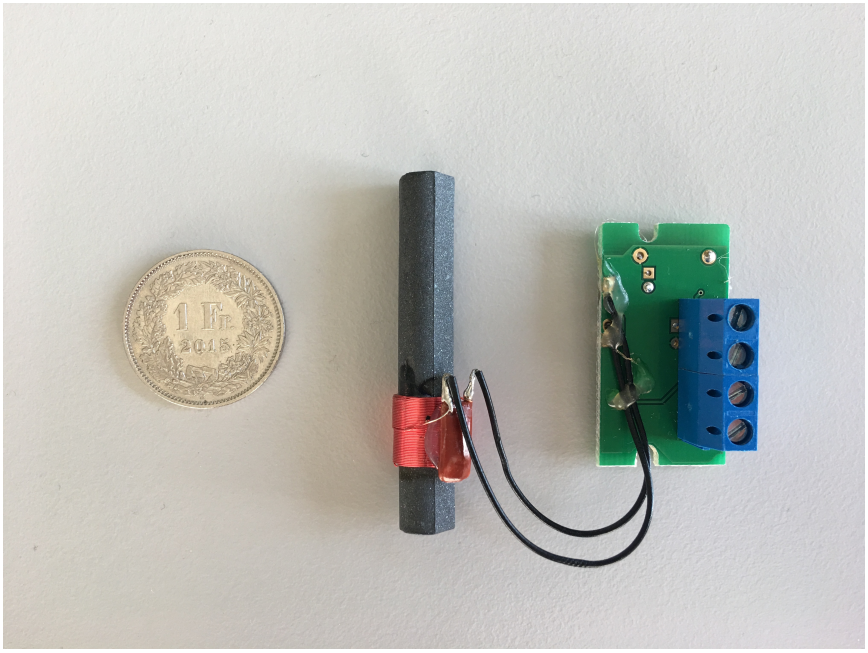


Figure 4.6: Conrad DCF77 receiver

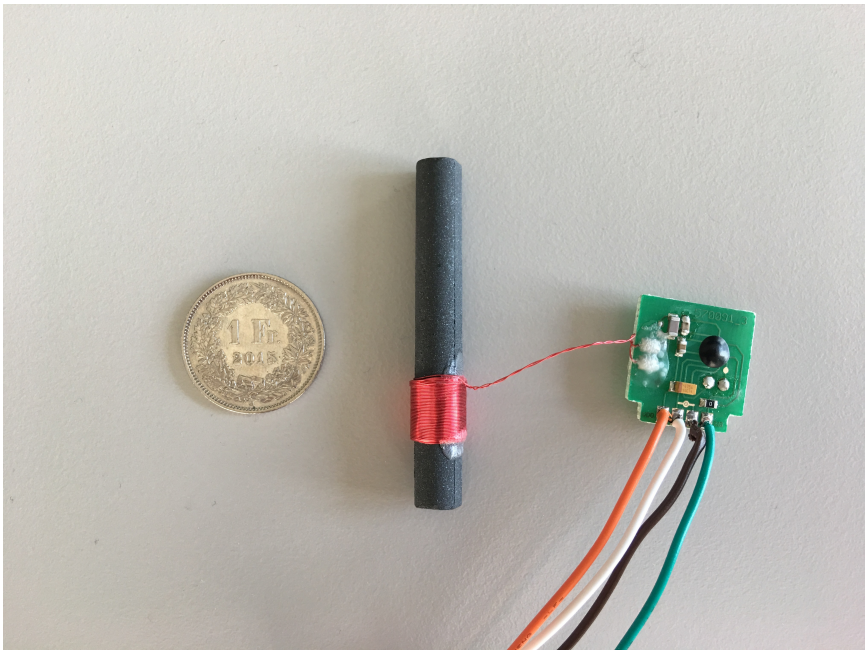


Figure 4.7: Reichelt DCF77 receiver

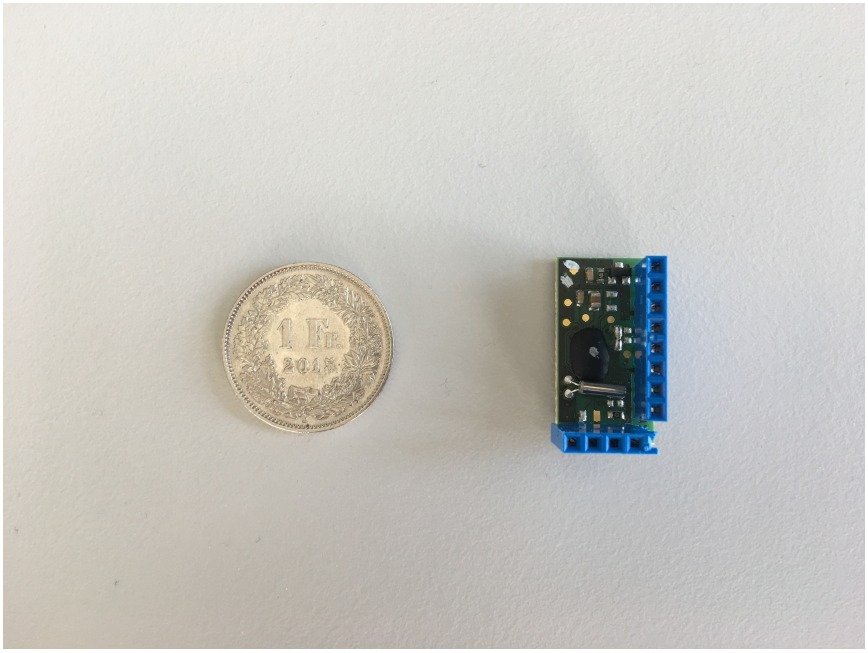


Figure 4.8: HKW DCF77 receiver without attached antenna

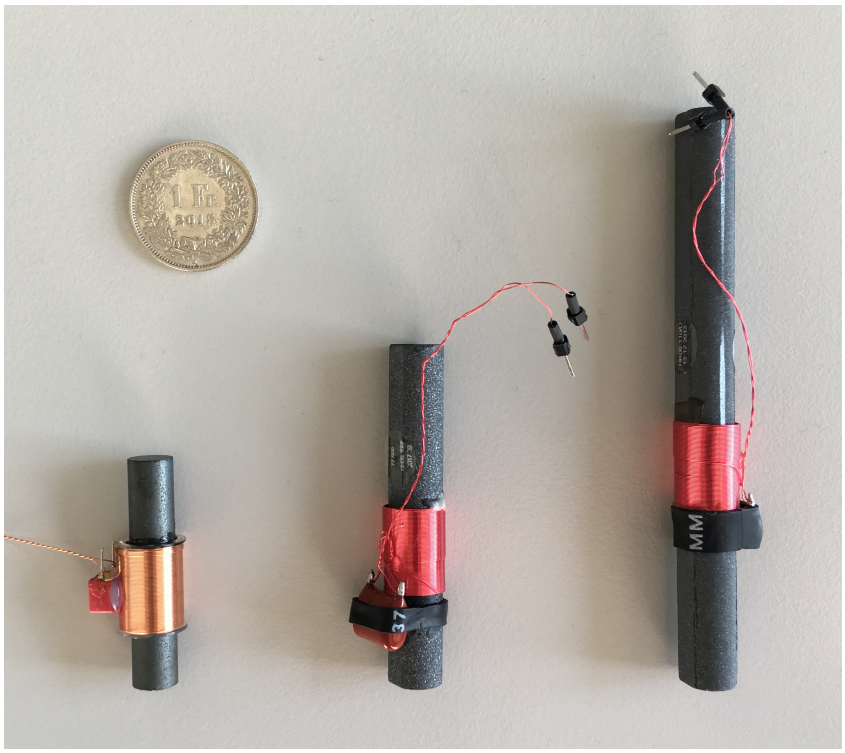


Figure 4.9: $40mm$, $60mm$, and $100mm$ antennas for the HKW receiver

4.2.3 Results and Discussion

Interference from LCD Screens

Placing the receiver close to any LCD screens, such as those found in laptops or cell phones, greatly affected the reception and quantization quality. The LED blinked erratically at a high frequency when the interference source was placed in the receiver's line of sight. Even when the screens were not placed directly in front of the receiver, a proximity of less than a meter caused erroneous behavior to be seen on the LED. This observation confirmed that proper function of the receiver is only achieved if no LCD screens are in close proximity. For reference, a sample comparison of the quantized DCF77 receiver output both far and close to LCD screens is shown in Fig. 4.10 and Fig. 4.11, respectively.

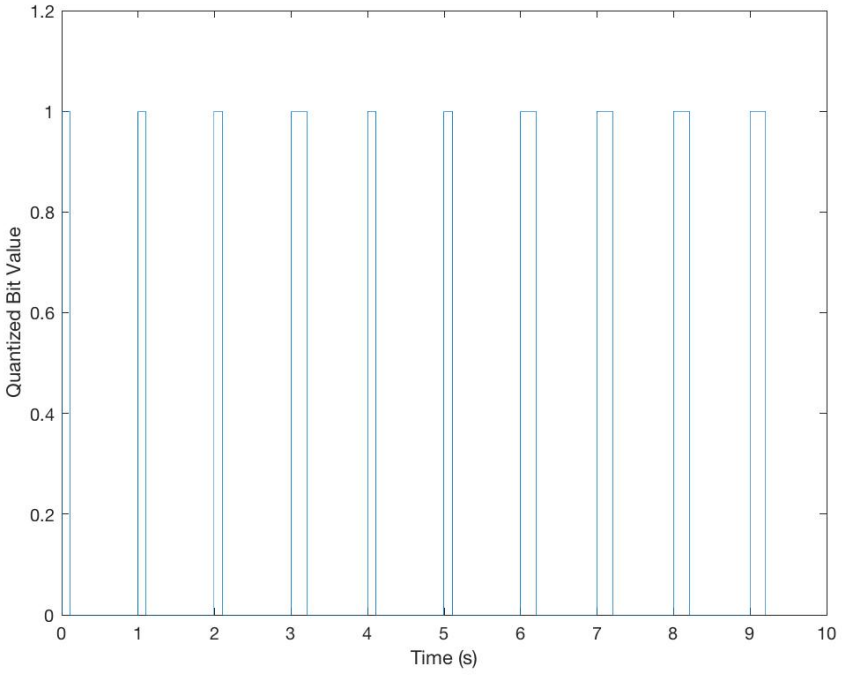


Figure 4.10: Reception of quantized DCF77 bits without interference

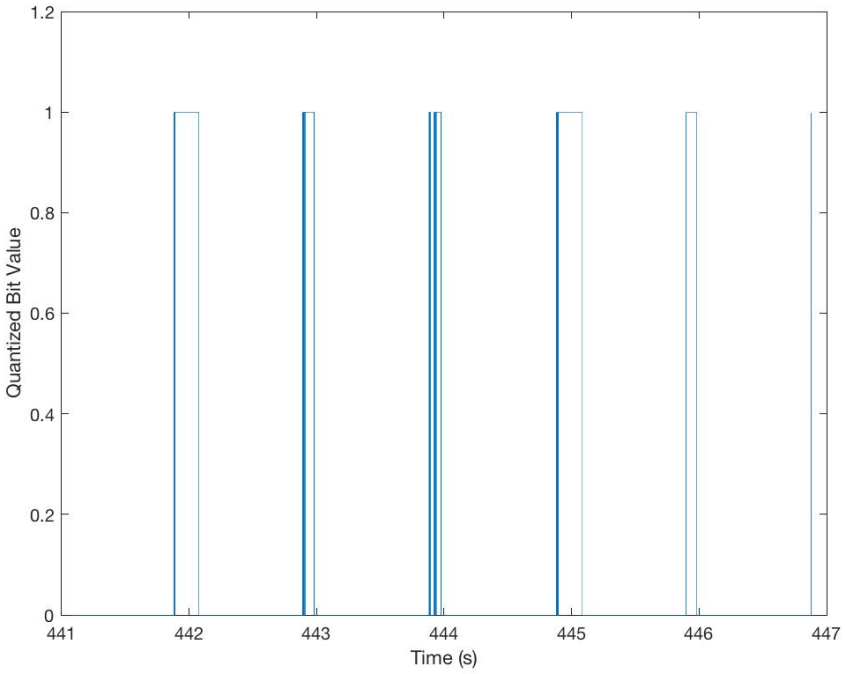


Figure 4.11: Reception of noisy quantized DCF77 bits in the presence of a cell-phone screen

Signal Blockage from Concrete Walls

The location of the DCF77 receiver in a concrete building did not have a large an effect on the reception quality as it did with the LCD screens. However, when the receiver was over five meters away from the closest window, the quality of reception decreased. The reason may not necessarily be due to more walls in the receiver's line of sight. Since the building in which the test was conducted is a densely packed apartment complex, many closely placed rooms were likely occupied by residents who also owned electronics, laptops, or computer monitors. Moving the receiver away from the window could have placed an adjacent room filled with electronics in the receiver's line of sight. As a result, the study showed that indoor applications would be limited depending on the proximity of nearby electronics.

Wall Socket Noise

In this experiment, the receiver was powered through a wall adapter that converted 50 Hz 220 Vrms into 5V DC, which was then regulated to 3.3V DC by a LT1086 linear regulator. In this particular setup, the DCF77 receiver did not quantize a single bit - the LED was completely off. The test was repeated for cases in which the 5V DC was provided by a USB port in a laptop that was powered by the wall connector. However, the LED still failed to show any quantized bits. However, when the DCF77's input voltage was provided by a battery source, such as a 9V battery, a standard portable cell phone charging battery, or even a laptop that was disconnected from the wall, the DCF77 receiver worked correctly. It appeared that any electrical connection made between the receiver and the wall, despite numerous decoupling capacitors placed on the DC voltage sources and near the receiver, would render the receiver dysfunctional.

Summary

Similar observations were made on all of the Conrad and HKW receivers. Both of the ordered Reichelt receivers did not function and one of them even had a short circuit as it was drawing excessive current. The tests were also performed by varying the size of the ferrite antenna used with the HKW receiver. There was no significant difference observed in the reception quality among the three antennas of different length. The results of the qualitative experiment are tabulated in Table. 4.2.

Test Number	Description	Impact
1	Interference from LCD screens	High
2	Signal blockage from concrete walls	Low
3	Wall socket noise	Very high

Table 4.2: Qualitative Tests for DCF77 Interference

The conclusions drawn from these qualitative tests provided insight into aspects that additional test setups must consider. The placement of testing equipment and how they are powered greatly affect the reliability of the DCF77 receiver.

4.3 Availability and Accuracy Study of DCF77

4.3.1 Concept

After investigating factors that affected the reception quality of DCF77, it was of interest to study the accuracy, precision, and availability of the DCF77 signal received through the low-cost receiver. Although DCF77 is generated by a high precision and high accuracy atomic clock source [5], the receiver could degrade the overall quality by introducing additional systematic or random timing errors, or even cause certain pulses to not be received. As a result, the receiver would limit the maximum usefulness of DCF77 as a synchronization source. The need to quantitatively understand how the receiver affected DCF77 motivated the following experiment, which studied the availability and accuracy of the quantized DCF77 signal.

4.3.2 Implementation

Since the previous test showed no noticeable differences between the HKW and Conrad receivers, one of the three Conrad receivers was arbitrarily selected to be used in all subsequent tests.

Determining the timing accuracy of the received DCF77 edges required a highly precise timer. To satisfy this requirement, the setup shown in Fig. 4.12 was used. In the setup, the ublox GPS receiver was configured to output a square wave of 8 MHz, which was used to clock a MSP430 Launchpad Development board. The MSP430 was then connected to the DCF77 receiver through one of its interrupt input pins in order to timestamp all received rising and falling edges. Timestamps of all rising and falling edges received by the MSP430 were stored in memory for post-processing.

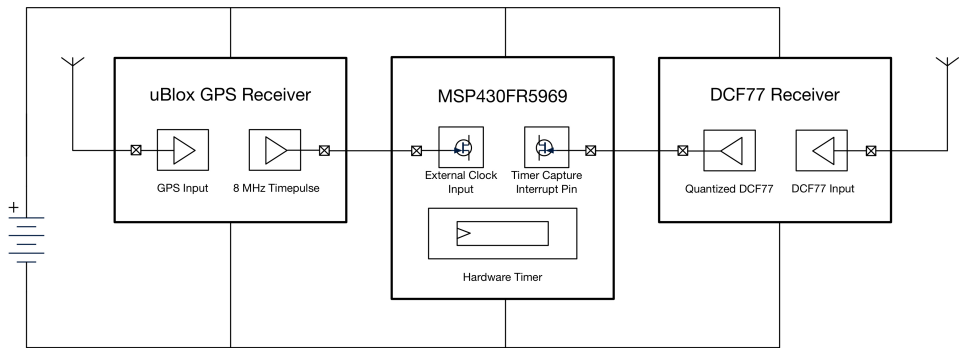


Figure 4.12: Block diagram of DCF77 accuracy experimental setup

4.3.3 Results and Discussion

The availability of DCF77 was studied by measuring the timestamps of all rising and falling edges outputted from the DCF77 receiver. The measured DCF77 data included timestamps of all received rising and falling edges over ten hours of testing. In the data, some falling edges followed too closely to rising edges, which was indicative of noise in the DCF77 receiver.

To analyze the data, the absolute difference between the timestamps of the rising edge and the following falling edge was calculated for each rising edge, which produced a set of pulse durations. Invalid durations, such as those that were shorter than 50 ms or longer than 250 ms, do not correspond to a bit value, so they were removed. The remaining pulses were then decoded into bits by categorizing all pulse durations between 50 ms to 150 ms as a bit 0 and all pulse durations between 150 ms to 250 ms as a bit 1. The resulting bits were then compared against archived DCF77 data [14] to further filter any bits that did not match the expected bit value in each frame.

DCF77 Availability

Since the filtered bits were compared against expected bits available from [14], the comparison allowed for knowledge on exactly which bits were received correctly, incorrectly, or not at all. The total number of received rising edges, decodable pulses, and valid bits are tabulated in Table. 4.3

Measurement	Value
Received Rising Edges	47651
Decodable Pulses	36088
Valid Bits	26963 (56.6%)

Table 4.3: Statistics on Received DCF77 Signal

The data was further analyzed by counting how many of the expected bits in each frame were received to determine the overall availability of the received DCF77 signal. The availability is shown as a percentage of correctly received bits in each frame over time in Fig. 4.13.

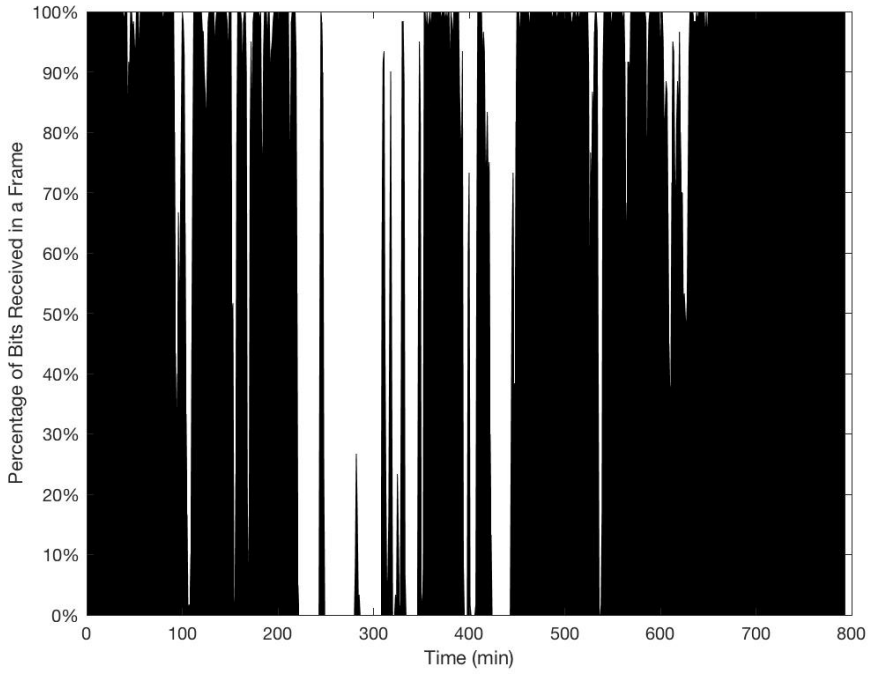


Figure 4.13: Percentage of correctly received bits in a frame over testing time

DCF77 Accuracy and Precision

Analysis of DCF77 accuracy and precision was done by measuring the time between consecutive rising edges. Since the DCF77 signal contains one bit of information every one second, consecutive rising edges were expected to be separated by exactly one second. However, some consecutive edges may be more than one second apart due to the DCF77 receiver missing certain pulses. In this case, the measured time difference was divided by the rounded number of seconds between them. The set of all the time differences are plotted in the histogram shown in Fig. 4.14. Statistics about the histogram are tabulated in Table. 4.4.

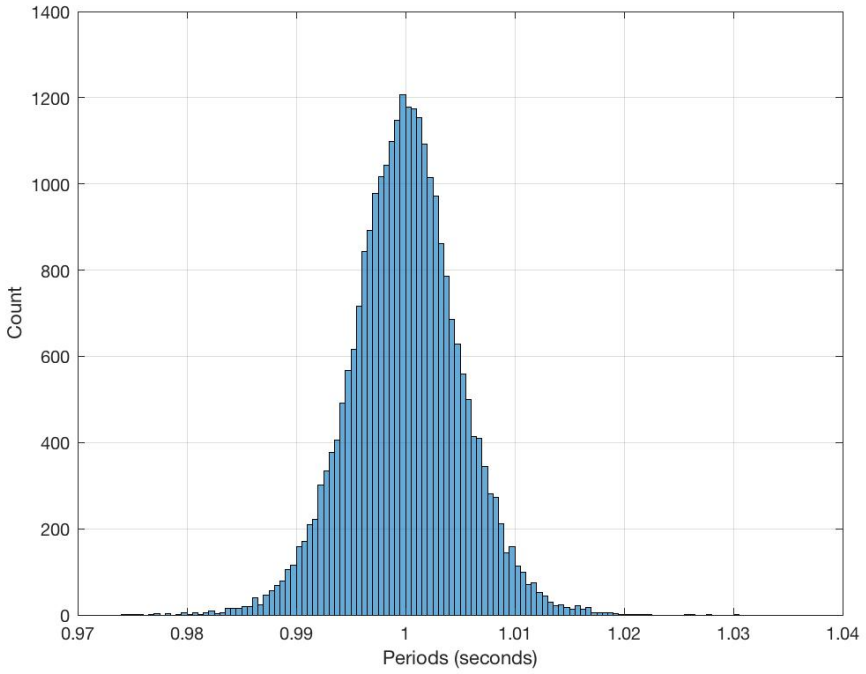


Figure 4.14: Histogram of periods between consecutive one second pulses

Statistic	Value
Mean Period (s)	0.99999586
Standard Deviation (s)	0.00500599
Mean frequency (Hz)	1.00000414

Table 4.4: Statistics of DCF77 accuracy determined from empirical data

It was also useful to understand the statistics concerning the accuracy and precision of the rising and falling edges separately, relative to a reference. Using the archived DCF77 bitstream from [14], the value of each bit was known. This information allowed a reference DCF77 signal to be recreated, which was used to compare against the measured DCF77 signal. Each rising edge from the measured signal was compared with the corresponding rising edge in the reference signal to determine whether it was early or late, and by how much. The time differences between the measured rising edge and reference rising edge are plotted in the histogram shown in Fig. 4.15. The comparison was also made for the falling edges.

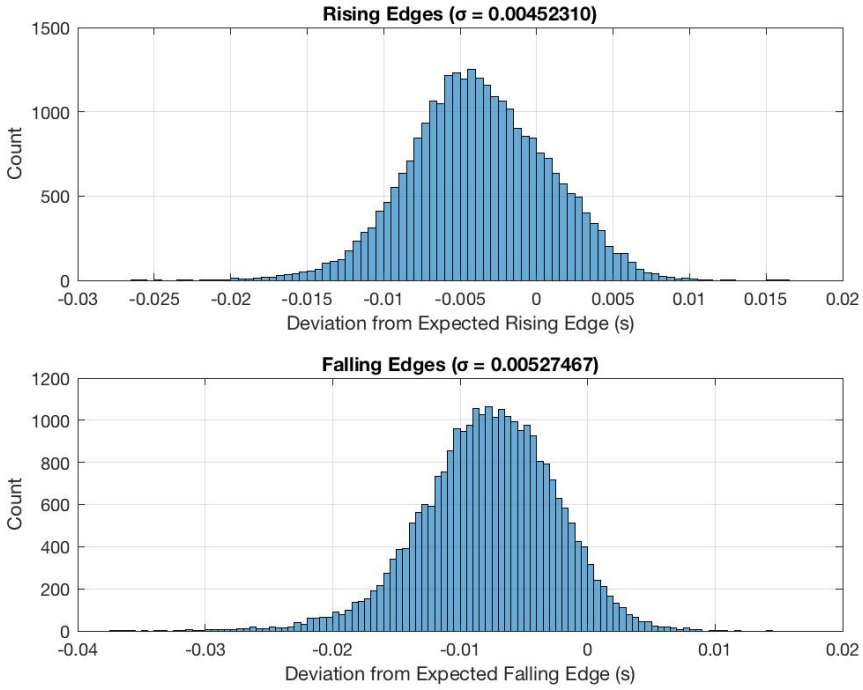


Figure 4.15: Histograms of rising and falling edge deviations from reference

Analysis of the rising and falling edges resulted in a standard deviation of approximately 0.005 seconds, which is comparable to the standard deviation of the histogram in Fig. 4.14, which plotted differences between consecutive rising edges. It is expected that these two standard deviations are similar because deviations in rising edges from the reference directly affects the time difference between adjacent rising edges, which is what the standard deviation from Fig. 4.14 was based on.

Analysis of the distribution for rising edge time differences resulted in a mean frequency of 1.00000414 Hz. Despite the calculated frequency being close to 1 Hz, the relative error was still larger than expected. This deviation was possibly due to both the inability of the DCF77 receiver to discern the start of bits with high precision, and the total number of data points used in this analysis.

For the DCF77 receiver to have precisely measured the sharp drop in amplitude of the DCF77 carrier wave, the receiver must have high bandwidth; or, at least as high as that of the transmitter and transmitting antennas of the source of DCF77. A low bandwidth antenna would have resulted in the edge being detected later by a slight delay. Furthermore, the DCF77 signal could have exhibited multipath effects due to the environment between the signal source and the receiver. The multipath would cause some waves to be received slightly earlier or later than

others, which would result in a wider distribution of the time difference between consecutive rising edges. Based on the standard deviation of $0.005s$ calculated from the histogram of the measured data, it was concluded that the unknown distribution was not very narrow given the accuracy specifications of DCF77. This implication meant that there was a small probability for some edges to be received much earlier or later compared to the mean. For the mean of the distribution to be close to unity, the distribution must be symmetric about unity. However, the wide distribution together with approximately only 27,000 data points, meant that even a few outliers could have easily shifted the mean. In this particular case, even five asymmetric data points at 1.02 could have resulted in the calculated mean. The number of accurate decimal places of the mean is improved as more data points are included in the study. As a result, this analysis has concluded that DCF77 exhibits frequency accuracy that is high and accurate enough to synchronize devices to the millisecond range. However, high accuracy does not imply high precision. In this case, the precision resulting from a standard deviation of $0.005s$ could pose potential problems to a DCF77 based timer device because it is in the same order of magnitude as the millisecond accuracy that needs to be achieved.

Chapter 5

Timer Device

The timer device, shown in Fig. 5.1, is a piece of hardware consisting of a microcontroller and a low cost DCF77 receiver. The timer device connects to host devices and is able to generate timed events with high accuracy due to synchronization with the received DCF77 signal.

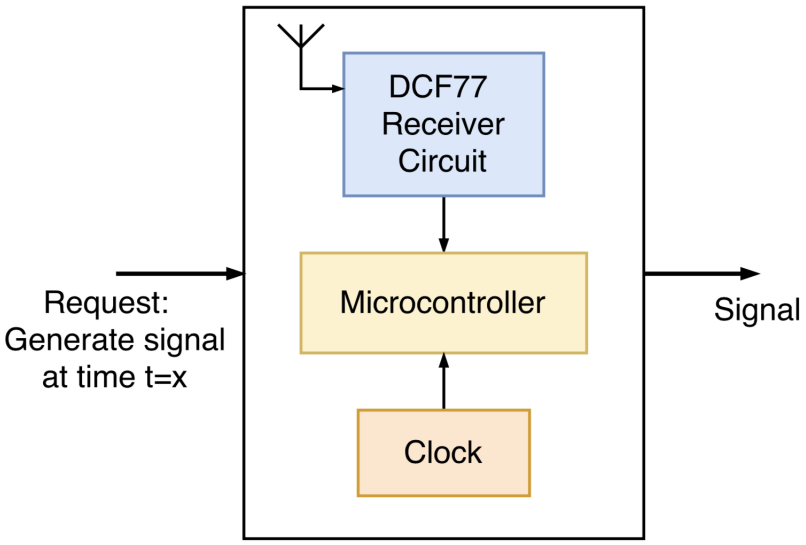


Figure 5.1: Block diagram of the timer device [1]

Requirements of the timer device were that it must be accurate and robust to temperature deviations and periods of DCF77 signal unavailability, as well as run for long durations on battery power. The timer device's synchronization algorithm needed to be developed to have an appropriate trade-off between accuracy and power consumption while satisfying all functional requirements.

5.1 Conceptual Design

The timer device should operate in low power sleep states as much as possible to reduce average power consumption. Since DCF77 pulses were received at a rate of at most one Hertz, the timer device should sleep during the one second interval in between received bits. The timer device was designed to have three operational modes (tabulated in Table. 5.1): awake mode, low power sleep mode, and ultra-low power sleep mode. During awake mode, both the microcontroller and the DCF77 receiver were powered on. During low power sleep mode, the microcontroller was in a standby sleep mode and not executing code, but was able to wake up from interrupts generated by the DCF77 receiver. Similar to in awake mode, the DCF77 receiver was also powered on during low power sleep mode. Note that the term "active mode" refers to union of the low power sleep and awake modes. Finally, in ultra-low power sleep mode, all parts of the microcontroller were powered off except for a RTC that was able to trigger an alarm to power on the microcontroller. The DCF77 receiver was powered off in this mode.

Operating Mode	Microcontroller	DCF77 Receiver
Awake	On	On
Low power sleep	Standby	On
Ultra-low power sleep	Off except for RTC	Off

Table 5.1: Timer device operating modes

This particular choice of operating modes was motivated by the need to reduce average power consumption over long durations as much as possible. Two different sleep modes were used to further reduce average power consumption for cases in which the timer device did not have to generate events for long durations.

During active mode, the DCF77 receiver was powered on and outputted quantized bits from the received DCF77 signal. As seen from the preliminary experiments on DCF77 availability, decoding an entire frame was not always possible, so the robustness of the algorithm may be degraded if it always required full frames to be correctly decoded. Therefore, the timer device limited decoding entire frames as much as possible and only worked with the individual pulses in the frames.

Let T_{TARGET} be the target time of the end event relative to the start event, T_p be the duration of time in which a full DCF77 frame can be received without error with at least probability p, and T_d be the minimum amount of time that could result in a clock drift of at least T_p. In Algorithm 1, the timer device immediately proceeds to ultra-low sleep mode for a duration of $\min(0, |T_{TARGET} - T_p|, T_d)$ so that when exiting the power mode (if it goes into ultra-low power sleep mode

at all), there is at least T_p amount of time remaining until the timer device has to generate the end event. Upon exiting the ultra-low power sleep state, the timer device synchronizes its internal time to CEST time by waiting to receive a full error-free DCF77 frame and then decode it. After which, the timer device corrects for subsequent clock drift by using received DCF77 pulses, which are known to have a difference of at least one second between consecutive pulses, until it is ready to generate the end event. The timer device continually transitions between the low power sleep and awake modes until the it has to generate the end event. During this time, the algorithm maintains a variable to track the frequency of its internal timer clock. The variable is updated by each subsequently received DCF77 pulse until the timer device has counted enough time to generate the end event.

Algorithm 1 Synchronization Algorithm, Variation 1

1: *Global variables*:
2: $T_{TARGET} \leftarrow$ user defined value
3: $T_p \leftarrow$ user calculated value based on desired value of p
4: $T_d \leftarrow$ user calculated value based on T_p
5: $internalClockFrequency \leftarrow$ nominal clock frequency
6: $internalTime \leftarrow 0$
7: $previousInternalTimestamp \leftarrow 0$
8: $secondsElapsedSum \leftarrow 0$
9:
10: **procedure** ENTRY POINT
11: setEndEventAlarmFromNow(T_{TARGET})
12: enterUltraLowPowerSleepMode(min($0, |T_{TARGET} - T_p|, T_d$))
13:
14: **procedure** EXIT ULTRA-LOW POWER SLEEP MODE CALLBACK
15: $CESTTime \leftarrow$ receiveAndDecodeDCF77Frame()
16: setEndEventAlarmFromNow($T_{TARGET} - (CESTTime - internalTime)$)
17: $interalTime \leftarrow CESTTime$
18:
19: **procedure** DCF77 RECEIVED PULSE INTERRUPT HANDLER
20: $currentTimestamp \leftarrow internalTime$
21: $difference \leftarrow currentTimestamp - previousInternalTimestamp$
22: $previousInternalTimestamp \leftarrow currentTimestamp$
23:
24: $secondsElapsed \leftarrow$ round($difference$)
25: $secondsElapsedSum \leftarrow secondsElapsedSum + secondsElapsed$
26: $internalClockFrequency \leftarrow (difference/secondsElapsed) * internalClockFrequency$
27: updateEndEventAlarm($secondsElapsedSum, internalClockFrequency$)

In this algorithm, the code maintains a variable to keeps track of the actual frequency of the microcontroller clock. On each received DCF77 edge, the variable is updated to reflect the new calculated value, which is determined by dividing the elapsed clock cycles by the estimated number of elapsed seconds. The updated frequency and elapsed seconds sum are then used in the updateEndEventAlarm function to modify the end event alarm accordingly. The algorithm is robust to missed DCF77 pulses because it measures the time between consecutively received pulses and divides that time by the rounded number of seconds between the pulses. This operation ensures that any missed pulses that the DCF77 receiver did not successfully quantize do not significantly impact the performance of the algorithm. Once the timer device has exited ultra-low power sleep mode, most of its time is spent in low power sleep mode. Only the interrupt handler executes in awake mode.

Although conceptually simple, this algorithm has a number of implementation issues that should be accounted for during design. For example, the DCF77 interrupt handler has delay that is accumulated before the call to the updateEndEventAlarm function, which cannot account for this delay in a straightforward way. The execution of the interrupt handler may take a non-deterministic amount of time depending on the processor architecture. The division operation between $difference$ and $secondsElapsed$ may be performed by a software routine if the processor does not have hardware support, which would add considerable overhead. Since the delay accumulates with every call of the interrupt handler, the accuracy of the timer device will deteriorate over time.

These imperfections can be reduced by designing the algorithm to measure, at run time, the interrupt handler delay and account for it. This motivated the design of Algorithm 2, which approaches the problem from the perspective of feedback control. By formulating the synchronization problem as a closed loop tracking problem in which the microcontroller corrects its internal clock frequency to be as close as possible to a desired nominal frequency, the results of feedback control can be applied. In feedback control (shown in Fig. 5.2), a dynamical system, called a plant, is affected by a control input, which is a function of the difference between a desired value of the system output and the actual output. This difference is known as the error. In the case of this synchronization problem, the plant is the internal time, which increments at a constant rate. The internal timer can be affected by a control input, which can increase or decrease the internal timer by any desired amount. The output of the system is the current frequency at which the internal timer increments, which is compared to the nominal frequency of the microcontroller's clock. The purpose of the controller is then to generate a control input so that the error becomes as small as possible, which is equivalent to forcing the internal timer's frequency to be as close as possible to the nominal value.

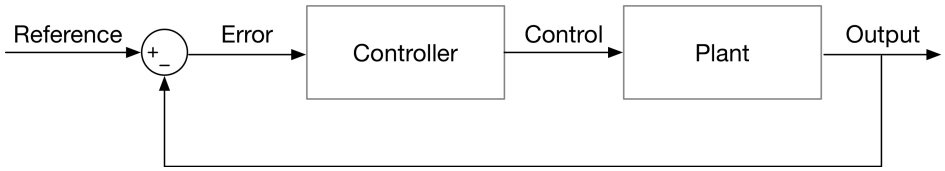


Figure 5.2: Feedback controller block diagram

Algorithm 2 Synchronization Algorithm with PD Controller (Variation 2)

Global variables:
  $T_{TARGET} \leftarrow$ user defined value
  $T_p \leftarrow$ user calculated value based on desired value of p
  $T_d \leftarrow$ user calculated value based on T_p
  $internalTime \leftarrow 0$
  $K_p \leftarrow$ PID proportionality constant
  $K_d \leftarrow$ PID differentiating constant
  $previousInternalTimestamp \leftarrow 0$

procedure ENTRY POINT
  setEndEventAlarmFromNow(T_{TARGET})
  enterUltraLowPowerSleepMode(min($0, |T_{TARGET} - T_p|, T_d$))

procedure EXIT ULTRA-LOW POWER SLEEP MODE CALLBACK
  $CESTTime \leftarrow$ receiveAndDecodeDCF77Frame()
  setEndEventAlarmFromNow($T_{TARGET} - (CESTTime - internalTime)$)
  $interalTime \leftarrow CESTTime$

procedure DCF77 RECEIVED PULSE INTERRUPT HANDLER
  $currentTimestamp \leftarrow internalTime$
  $difference \leftarrow currentTimestamp - previousInternalTimestamp$
  $previousInternalTimestamp \leftarrow currentTimestamp$

  $error \leftarrow$ round($difference$) $- difference$
  $input \leftarrow K_p * error + K_d * (error - previousError)$
  $internalTime \leftarrow internalTime + input$
  setEndEventAlarmFromNow($T_{TARGET} - internalTime$)

In comparison to the first algorithm, the second one has internal state that accounts for the execution time of the DCF77 interrupt handler. As a result, the overhead delay of the interrupt handler is handled by the closed loop control, so it will not accumulate on every call of the interrupt handler. Furthermore, by directly modifying the internal time by adding the control input, the division operation from line 26 in Algorithm 1 is avoided. To gauge the performance of this algorithm, a simulation was first performed. The algorithm was implemented in MATLAB and then ran against actual DCF77 data that had been collected in the preliminary DCF77 availability experiments. The errors associated with a timer device running the synchronization algorithm and a control timer device that does not run any synchronization are plotted against different clock frequencies centered around the nominal 32.768 KHz in Fig. 5.3.

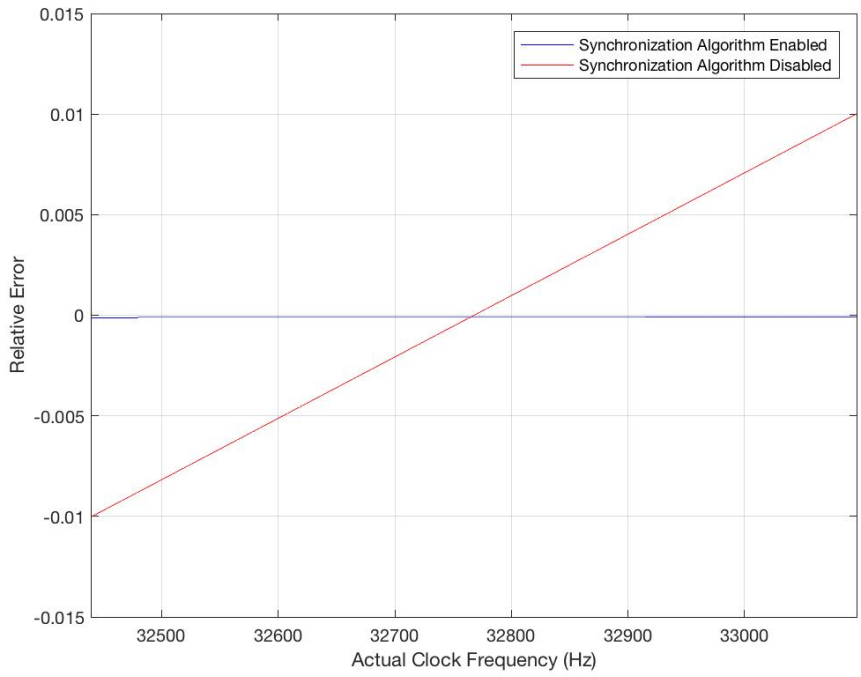


Figure 5.3: Simulated reduction of clock drift error using synchronization

5.2 Implementation

5.2.1 Mapping Power Modes

The MSP430FR5969 was chosen as microcontroller to be used in the timer device. The many low power modes of operation of the MSP430FR5969 simplified the mapping of the timer device power modes described in Table. 5.1 to power modes of the microcontroller. Shown in Table. 5.2, the awake power mode was mapped to the AWAKE mode of the MSP430; the low power sleep to the LPM3 mode, and the ultra-low power sleep to LPM3.5. In the AWAKE mode, all initialized peripherals in the MSP430 are powered and the MSP430's CPU runs at a frequency of 1 MHz. In LPM3 mode, only the MSP430's CPU is powered off, but all peripherals remain on. In LPM3.5 mode, everything except for the MSP430's RTC is powered off. As described in the timer device's conceptual design, the DCF77 receiver is also powered off during ultra-low power sleep mode.

Operating Mode	MSP430 Power State	Description
Awake	AWAKE	Everything is powered on
Low power sleep	LPM3	CPU is powered off
Ultra-low power sleep	LPM3.5	Only RTC is powered on

Table 5.2: Timer device operating modes

The transitions between the three power modes as dictated by Algorithm 2 can be modeled using a state transition diagram shown in Fig. 5.4. In this diagram, the three states correspond to the three mapped power states in the MSP430.

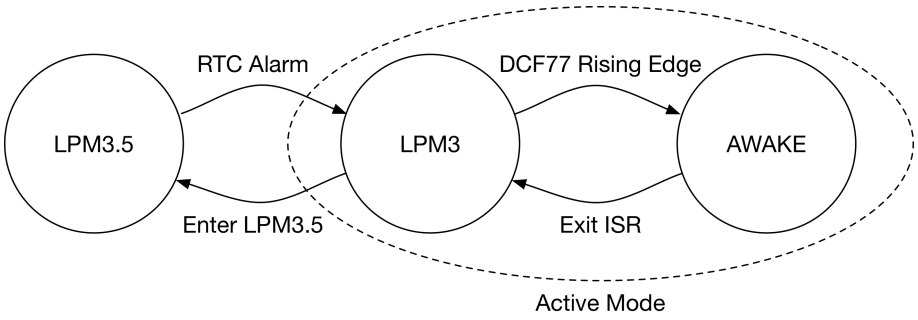


Figure 5.4: State transition diagram of the synchronization algorithm

5.2.2 Hardware Timer

To realize the synchronization algorithm on the MSP430 hardware, a number of challenges had to be overcome. Limitations of the MSP430, such as the lack of 32-bit hardware timers, required other methods to achieve the same level of performance. Since the timer device needed to be able to record time for up to 24 hours, a 32-bit timer running at the nominal crystal frequency of 32.768 kHz was sufficient. Note that only the hardware timer in the MSP430 ran at 32.768 kHz during active mode. The MSP430's CPU runs at 1 MHz when it's on, and 0 Hz when the timer device is not in awake mode. In the MSP430, only 16-bit timers were available, so two 16-bit timers were cascaded to simulate a single 32-bit timer. The issue with this implementation was that writing a new value to the 32-bit timer was no longer atomic, since two 16-bit words had to be written separately. Because the timer was simultaneously being updated by the main system clock, the non-atomic write could have led to race conditions. Fortunately, this implementation difficulty was overcome by pausing the timer each time it was to be written by software. The extra delays incurred by the pausing were automatically accounted for by the closed loop control strategy in Algorithm 2, which combined all delays in the synchronization computations as part of the error signal.

5.2.3 Ignoring Noise

As seen from the previous experiment on DCF77 availability, sometimes the DCF77 receiver outputted additional edges as noise. These additional rising edges would harm the accuracy of the timer device, which assumed that consecutive rising edges would be at least one second apart. To correct for this issue, the input interrupt trigger to the MSP430 was disabled for 30000 clock cycles, which is approximately 980 milliseconds at a clock frequency of 32.768 kHz, after each received rising edge. As a result, noisy bits during this interval do not trigger the MSP430's interrupt handler, and so are ignored. The implementation difficulty of this strategy was that to re-enable the input interrupt trigger after 980 milliseconds, the MSP430 had to be in AWAKE mode for the CPU to be active in order to write the appropriate switch in the memory-mapped port controller. This requirement would have been a significant power consumption performance hit. To overcome this difficulty, the direct memory access (DMA) controller was configured to automatically write the input interrupt trigger enable switch to the memory-mapped port controller whenever the 980 milliseconds timer reached its target count of 30000. This strategy meant that the CPU inside the MSP430 did not have to leave LPM3 mode, so additional power consumption penalties were not incurred.

5.3 Performance Evaluation

5.3.1 Accuracy Validation

To characterize the timing accuracy of the timer device, a test bench was developed. The test bench was a second MSP430 Launchpad Development board running at 8 MHz with a clock input that was sourced from the ublox GPS receiver timepulse output. The test bench evaluated the accuracy of the timer device by measuring the time difference between the start event that it sent to the timer device and the end event received from the timer device. Since the timer device was preprogrammed to generate the end event after a known amount of time, the time differences measured by the test bench could be directly compared to expected results.

Accuracy with Perfect Input

To first determine the performance of the timer device under perfect input, the test bench was also designed to output an artificially generated pulse-per-second signal. Shown in Fig. 5.5, this signal connected to the quantized DCF77 input of the MSP430 (normally connected to the output of the DCF77 receiver) in order to allow the test bench to measure the accuracy of the timer device's synchronization algorithm with perfect input.

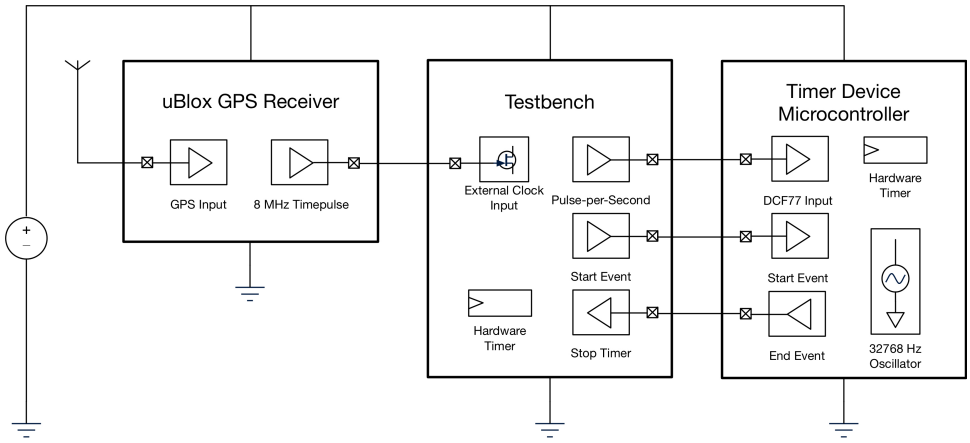


Figure 5.5: Block diagram of test for evaluating timer device accuracy

For this measurement, the test bench would trigger a start event with a rising edge to the input of the timer device while starting a timer. Once the timer device counted to the required output event time, it would trigger the end event with a rising edge that simultaneously stopped the timer in the test bench and

concluded the trial in the test. Different durations ranging from one minute to eight hours were chosen to observe if errors would accumulate with time. The results of the different durations are tabulated in Table. 5.3, and plotted in Fig. 5.6.

Duration (mins)	Expected Cycles	Measured Cycles	Difference (μs)
1	480,000,000	479,999,665	41.9
2	960,000,000	959,999,744	32.0
5	2,400,000,000	2,399,999,604	49.5
10	4,800,000,000	4,799,999,879	15.1
15	7,200,000,000	7,199,999,690	38.8
30	14,400,000,000	14,399,999,532	58.5
60	28,800,000,000	28,799,999,960	5.0
120	57,600,000,000	57,599,999,743	32.1
240	115,200,000,000	115,199,999,698	37.8
480	230,400,000,000	230,399,999,874	15.8

Table 5.3: Timer device with perfect PPS input accuracy test results (8 MHz test bench)

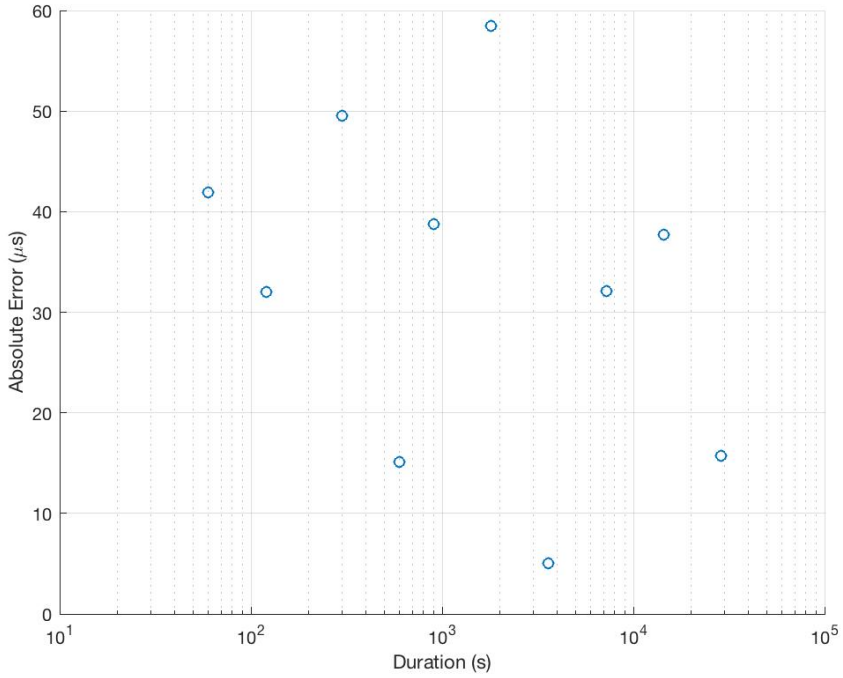


Figure 5.6: Relative accuracy error over testing duration

From the observed results, it appears that the accuracy error between the expected test bench count and the measured count is not affected by testing duration. This result implies that the error is not accumulating with the testing duration, as expected from the feedback closed loop design of the algorithm. It is interesting to note that even though 32.768 kHz crystal clocking the hardware timer was measured in a preliminary study to be running at 32.766 kHz, the same accuracy was achieved for all testing durations. The results of this experiment imply that any additional errors from future runs of the test would only be from the quantized DCF77 input to the timer device.

Accuracy with DCF77 Input

A final accuracy experiment was performed in which the MSP430 sourced its synchronization input from the DCF77 receiver instead of the artificially generated pulse-per-second output of the test bench. All other aspects of the setup were identical to the Accuracy with Perfect Input test. This experiment was more representative of a real application of the timer device. A block diagram of the experimental setup is shown in Fig. 5.7.

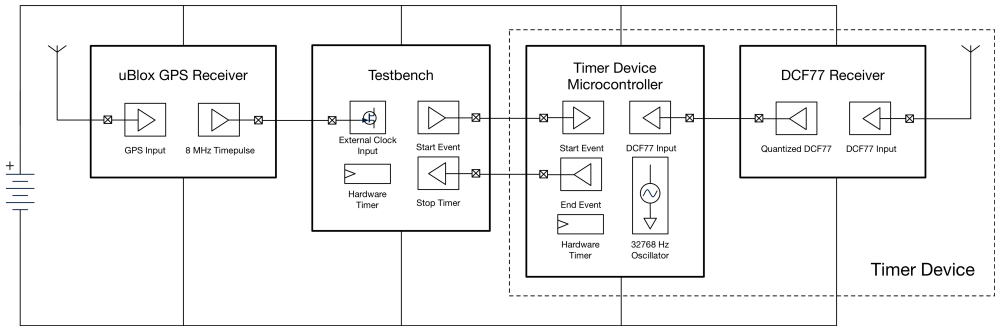


Figure 5.7: Block diagram of timer device using DCF77 as a synchronization source

In this experiment, the timer device was requested to generate an end event after 7 hours. The expected number of elapsed clock cycles, the measured number, and the absolute time difference are tabulated below in Table. 5.4.

Duration (mins)	Expected Count	Measured Count	Difference (s)
420	201,600,000,000	201,589,965,190	1.25

Table 5.4: 7 Hour experimental results for DCF77 sourced timer device (8 MHz test bench)

The inaccuracies seen from the testing results could have possibly been caused by two factors: the precision of the time difference between consecutive DCF77 rising edges, and a possible race condition in asynchronously reading the internal timer of the MSP430.

As seen from the DCF77 Accuracy and Precision section, the duration between consecutive rising edges of the DCF77 signal had a distribution approximated by the histogram in Fig. 4.14. The duration was centered around a mean value of $0.99999586s$ and had a standard deviation of $0.00500599s$. The mean is a measure of how accurate the duration was to exactly $1s$, whereas the standard deviation is a measure of precision. In this case, the standard deviation of $0.00500599s$ meant that the timer device's synchronization algorithm received around 95% of all pulses to be within plus-minus two standard deviations, or $1.00 \pm 0.01s$, assuming an approximate normal distribution. Since the durations had statistical spread, it is expected that the accuracy of the timer device's synchronization algorithm also had resulting spread. The deviation of $1.25s$ in 25200 seconds of testing corresponds to a relative deviation of approximately 0.00005, which falls within one standard deviation of the rising edge difference. If the timer accuracy experiment were repeated many times, the distribution of the deviations would likely be similar. The precision of the DCF77 receiver in outputting edges that were as close as possible to a set of ideal DCF77 edges could explain the reason for the deviation in the measured results.

Another possible reason for the inaccuracy of the timer device could have resulted from the asynchronous capture mode of the MSP's timer. In the implementation, the MSP used a 32-bit timer that was created by cascading two 16-bit timers such that the overflow output of one timer clocked the second timer. Capturing a timer value was done asynchronously because the second 16-bit timer only updated when the first 16-bit timer overflowed. According to the MSP datasheet, asynchronous captures may result in a race condition in reading a timer value that was simultaneously being updated [10]. As a result, the read value could be smaller or larger by any amount, depending on which bits inside the timer were being updated. If a single value was read to be smaller by 40960, or $1.25s$ at 32.768 kHz, it would cause the synchronization algorithm to incorrectly advance the internal timer by $1.25s$ in order to compensate, resulting in the timer device generating the end event $1.25s$ too early. Although the race condition is rare, for a single occurrence in 26200 seconds does not seem unlikely.

5.3.2 Power Characterization

Since the different DCF77 receivers all provide a similar interface to the microcontroller, they were interchangeable from a functional perspective. The modularity of the timer device allowed different DCF77 receivers to be used without significant modification to the hardware. Therefore, it was of interest to evaluate the power consumption performance of each modular part separately to better understand how power was consumed in the overall timer device. In the first part of this experiment, several power measurements were performed to exercise various permutations of the receivers and antennas. In the second part, the power performance of the entire timer device was characterized.

Measurement Setup

To measure the power consumption over time, the RocketLogger measurement unit was used. The RocketLogger had both a voltage channel and a current channel connected to the device under test. By measuring both the voltage and current over time, the power consumption could be determined. The measurement was performed for tests that involved only a DCF77 receiver and tests that included the complete timer device (MSP430 and DCF77 receiver). Fig. 5.8 shows a block diagram of the RocketLogger measuring a DCF77 receiver only. In Fig. 5.9, the entire timer device is under test.

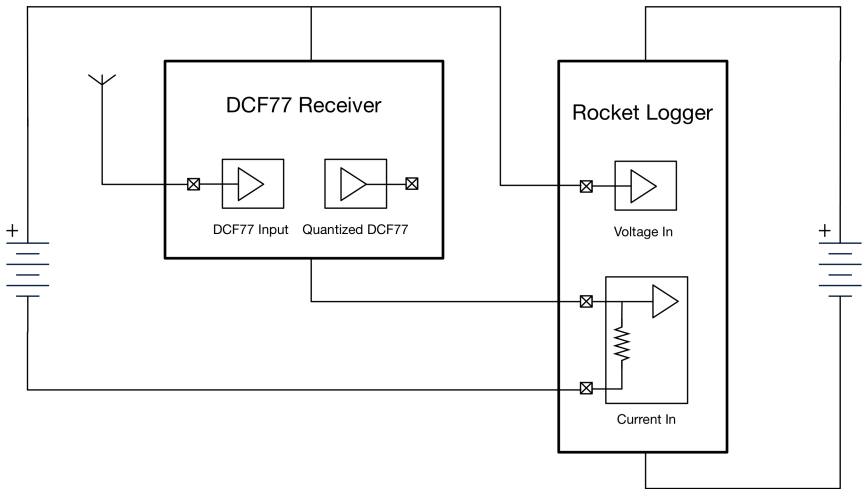


Figure 5.8: Power measurement of DCF77 receiver only

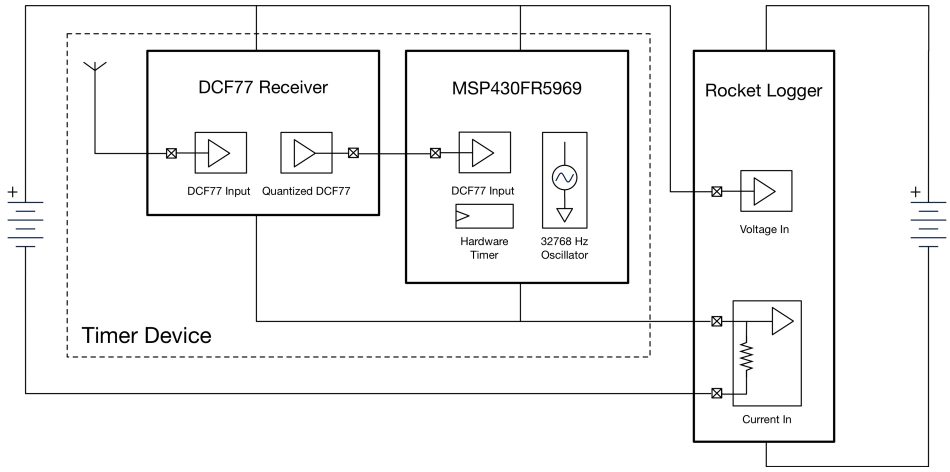


Figure 5.9: Power measurement of DCF77 receiver with MSP430 microcontroller

For measurements that were performed on only the DCF77 receivers, a total of three Conrad receivers, two Reichelt receivers, and one HKW receiver with three different lengthed antennas were tested. Letters are used to distinguish between the different instances of the same receiver. Measurements were also performed on the timer device, which used the best performing DCF77 receiver in terms of power. Measurements of the timer device in the active and ultra-low power sleep modes were performed separately due to the large discrepancy between the levels of consumed power. The overall test plan is shown in Table. 5.5.

Test Number	Configuration
1	Conrad A
2	Conrad B
3	Conrad C
4	Conrad B with 9.903kΩ pull-up resistor
5	Reichelt A
6	Reichelt B
7	HKW with 100mm antenna
8	HKW with 60mm antenna
9	HKW with 40mm antenna
10	Timer device in Active mode
11	Timer device in ultra-low power sleep mode

Table 5.5: Test plan of power characterization for different DCF77 receivers

DCF77 Receiver Results

The RocketLogger was connected to each DCF77 receiver independently and measured timeseries current and voltage data for a duration of ten minutes. The test was also repeated for one of the Conrad receivers with a pull-up resistor of 9.903kΩ attached to the output of the module. The pull-up resistor was necessary for correct functionality, but different values of resistance were tolerable. Therefore, it was omitted for comparison tests and only included in a single test to show the additional power requirements that it added to the receiver.

Fig. 5.10 shows a sample plot of the power consumed by the Conrad B receiver during ten seconds of test time. The plot was generated by a point-wise multiplication of the voltage and current timeseries data. In the plot, the pulses correspond to the received DCF77 pulses. The narrow pulses represent bit 0 and the wide ones represent bit 1. During the 7 second mark, a pulse is missing. Most likely, that is the 59th bit, which is always missing to indicate the start of a new DCF77 frame. From this plot, the peak and average power consumption can be easily determined by taking the maximum and sum of the power timeseries data, respectively.

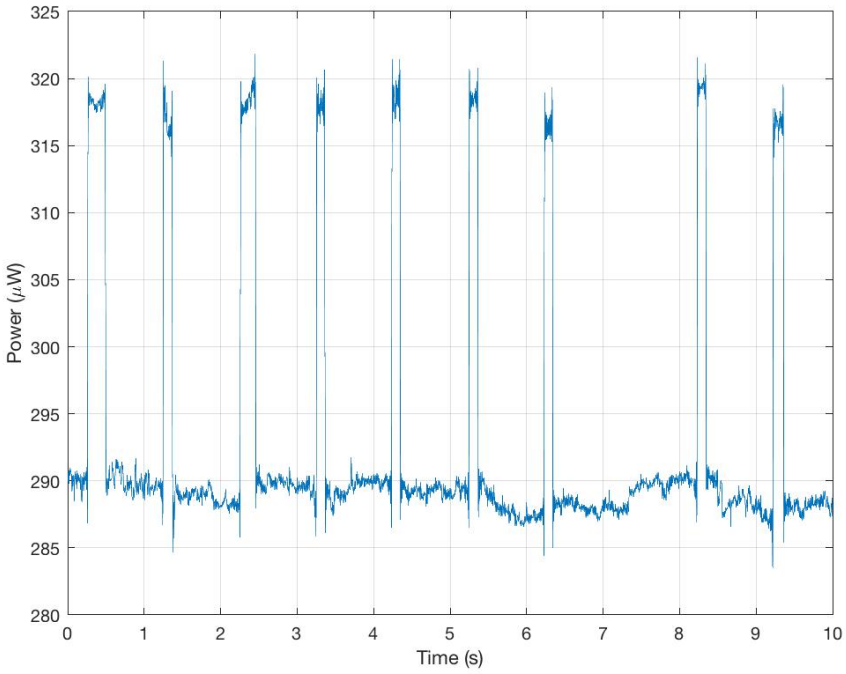


Figure 5.10: Sample power consumption plot of Conrad receiver B

Fig. 5.11 shows a power consumption plot of the same DCF77 receiver with a 9.903kΩ pull-up resistor connected. The pull-up resistor is connected to the active-low output of the DCF77 receiver, so that a low output pulse from the receiver corresponds to a high pulse in the consumed power. The active-low output means that when the output is active, current is dissipated through the resistor, resulting in the high pulse of the consumed power. Observation of the measured values reveals that the pull-up resistor increases the height of the pulses by over $1mW$. For a timer device that needs to be low powered, the extra power consumed by the pull-up resistor is significant.

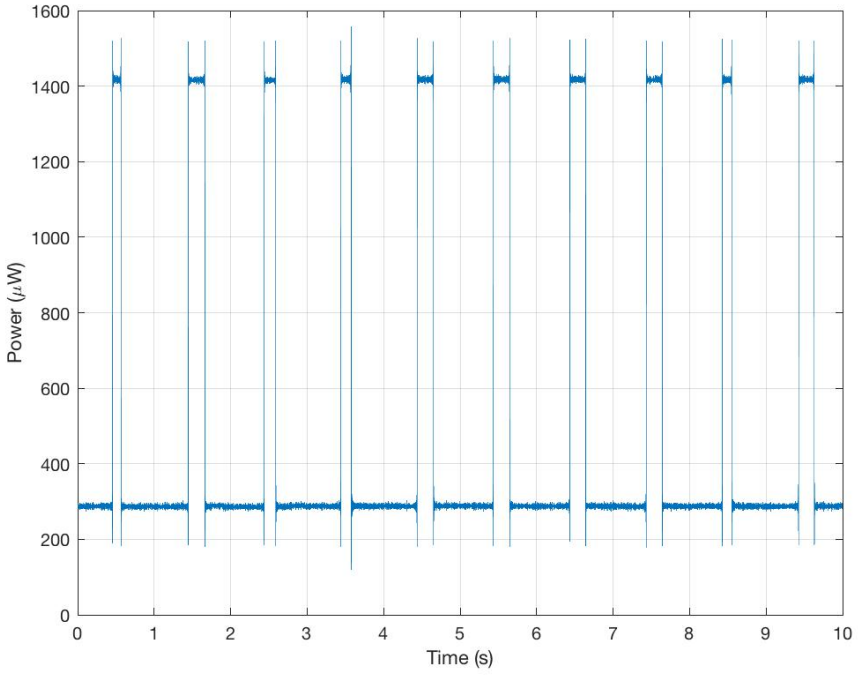


Figure 5.11: Sample plot of Conrad receiver B with 9.903kΩ pull-up resistor

Testing results of all the receivers are presented in Table. 5.6. Of the tested receivers, both of the ones ordered from Reichelt were faulty and so they were omitted from further testing. Three of the receivers ordered from Conrad appeared to be identical; however, their power consumption varied greatly among themselves. Most likely due to low-cost PCB fabrication variances, the differences in average power consumption were the cumulative effects of variance among hobbyist-grade electrical components. Conrad receiver "B" performed best in average power consumption, so it was used for subsequent full system tests. There were no noticeable differences in power consumption for the HKW receivers with different sized antennas.

Configuration	Average (μW)	Peak (μW)
Conrad A	1265.335	1302.783
Conrad B	293.174	326.132
Conrad C	400.563	430.100
Conrad B with pull-up resistor	463.800	1715.240
HKW with large antenna	1269.451	1654.373
HKW with medium antenna	1263.947	1723.206
HKW with small antenna	1279.189	1615.035

Table 5.6: Power characterization of different DCF77 receivers

Timer Device Results

Since the DCF77 receivers were functionally similar, the one with the best power performance (Conrad B) was chosen for all subsequent tests. With the MSP430 connected to the DCF77 receiver and running Algorithm 2, the timer device's current and voltage were measured by the RocketLogger for ten minutes. In the sample plot shown in Fig. 5.12, the peaks correspond to the timer device in active mode and current being dissipated through the pull-up resistor. Correspondingly, the region between the peaks corresponded to LPM3 low-power sleep state.

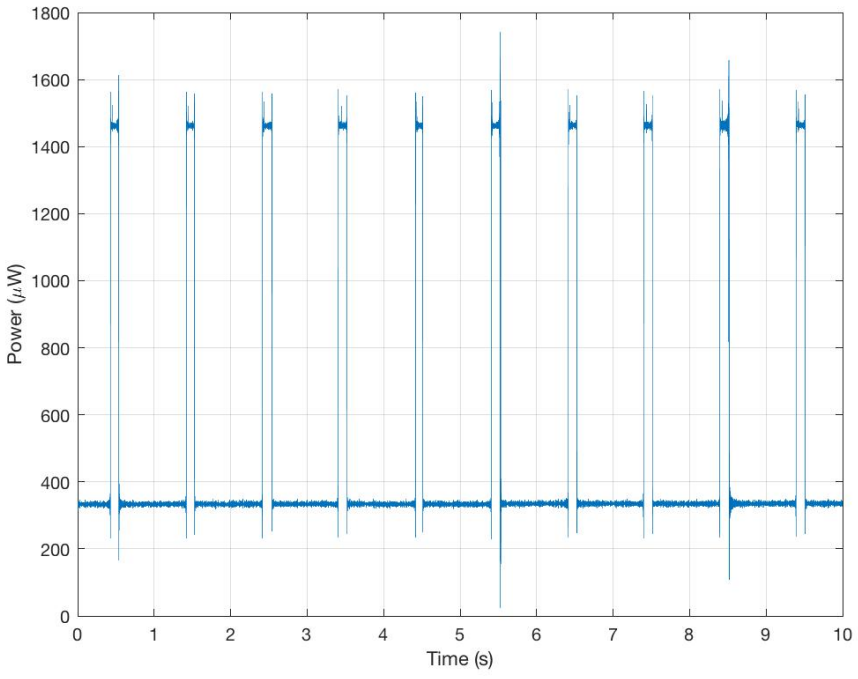


Figure 5.12: Sample plot of power consumption of entire timer device

Since the timer device operates in regular sleep and awake modes (LPM3 and AWAKE), as well as ultra-low power sleep mode (LPM3.5), it was important to characterize power consumption for all of these modes. The ultra-low power sleep mode is entered initially in Algorithm 2. In the sleep state, only the internal RTC was enabled, which is responsible for generating a CPU wake-up interrupt to exit LPM3.5. The expected power consumption levels for LPM3.5 is around $1.485\mu W$ [9], which is two to three orders of magnitude lower than those of LPM3.

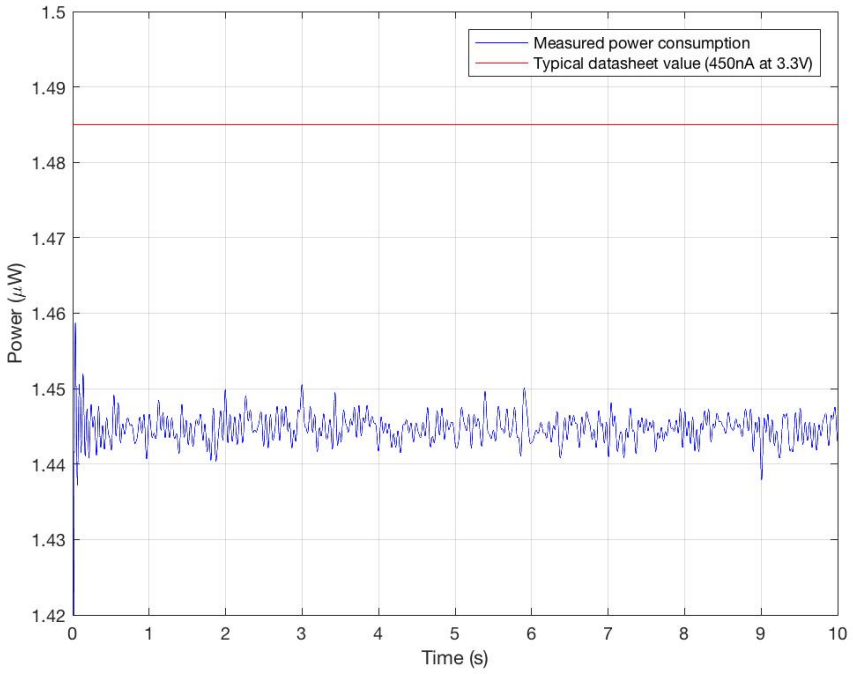


Figure 5.13: Sample plot of timer device in ultra-low power (LPM3.5) mode power consumption

In Fig. 5.13, both the measured power consumption and the nominal datasheet value are plotted for a sample duration of ten seconds. In this plot, it appears that the datasheet value at 25°C is an overestimate for the actual consumption, but values that are this low can easily be influenced by process variation and systematic uncertainties.

After the timer device was fully characterized in each of its power states, an equation for estimating the total average power consumption parameterized by the fraction of time spent in LPM3.5 can be realized.

$$P_{TOTAL} = \alpha P_{LPM3.5} + (1 - \alpha) P_{ACTIVE} \tag{5.1}$$

The timer device is directly affected by the α parameter. Being able to spend more time in the ultra-low power sleep mode LPM3.5 will result in lower overall average power consumption.

5.4 Discussion

The timer device was developed to use DCF77 as a synchronization source in order to provide time synchronization capabilities for wireless IoT devices without consuming channel capacity or significantly affecting power consumption. The timer device was developed around a low-cost DCF77 AM receiver, which was able to output DCF77 edges that a microcontroller could read. The timer device also included a MSP430 microcontroller that ran the synchronization algorithm, which was responsible for generating an end event after a preset duration following the reception of a start event. The timer device slept in an ultra-low-power sleep state for a duration of time long enough so that when it woke up, there was still enough remaining time, T_p, to receive a full frame of DCF77 with probability, p. The timer device also did not remain in ultra-low-power sleep mode long enough to allow the clock to drift enough so that the total time exceeded the preset duration, T_{TARGET}. From the DCF77 availability studies, 57.5% of full DCF77 data frames were received. If this percentage is taken as the probability with which a single frame can be correctly received, then the number of minutes required for at least one DCF77 frame to be received and decoded with at least 99% probability is calculated below.

$$T_{99\%} = \left\lceil \frac{\log(1 - 0.99)}{\log(1 - 0.575)} \right\rceil = 6 \tag{5.2}$$

From the temperature dependency of clock frequency study, the actual frequency could deviate from the nominal by as most as 0.5% if the temperature change is large enough. From these results, the timer device can remain in ultra-low-power sleep state LPM3.5 for at most:

$$T_{LPM3.5} = \min\left(0, |T_{TARGET} - 6 \text{ minutes}|, \frac{T_{TARGET}}{1.005}\right) \tag{5.3}$$

Once the timer device exits LPM3.5 mode, it will synchronize its internal timer to CEST time by decoding the next available DCF77 frame. For the remaining time, the timer device will operate in active mode, during which it relies on measuring the time between consecutive DCF77 edges to achieve accurate time keeping. Finally, the end event is generated once the target time has been reached.

The total power consumption of the timer device is parameterized by how much time it spends in active mode. To minimize overall consumption, the timer device should spend as much time in LPM3.5 as possible. The amount of time spent in LPM3.5 is $T_{LPM3.5}$, which is given by equation 5.3 above. Using equation 5.3 for $T_{LPM3.5}$, the average overall power consumption is then given by the following equation.

$$P_{TOTAL} = \left(\frac{T_{LPM3.5}}{T_{TARGET}}\right) P_{LPM3.5} + \left(1 - \frac{T_{LPM3.5}}{T_{TARGET}}\right) P_{ACTIVE} \tag{5.4}$$

Chapter 6

Conclusion

6.1 Future Work

Over the course of this semester thesis, the timer device was designed, developed, and tested after independent studies on the temperature dependency of microcontroller clocks and the accuracy and reliability of the DCF77 time signal. The timer device was designed to synchronize to DCF77 while prioritizing low power consumption without sacrificing time keeping accuracy. Throughout the studies and developments, several limitations and difficulties were encountered that could be overcome in future studies.

One such limitation encountered was the inability of the low-cost DCF77 AM receivers to receive edges with a delay that had a narrow distribution. Since edge detection in AM receiver circuits depend on antenna bandwidth and accurate component impedances, a low-cost receiver is unlikely to achieve very high performance. As a result, the limitation resulted in inaccuracies in time measuring, which resulted in sub-optimal performance of the timer device. Possible areas of future investigation are to explore more dedicated AM receiver circuits that can discern the DCF77 edges with less variance, and to consider the usage of DCF77's phase modulation scheme to achieve higher accuracies. Both methods would potentially result in increased accuracy at the expense of power consumption. Depending on the application, such a trade-off may be justified.

Another limitation encountered during this semester thesis was the lack of a 32-bit timer in the microcontroller. The particular line of MSP430 microcontrollers was due to its low power capabilities and availability at the time of this project. However, such a choice meant that a trade-off had to be made between functional hardware modules and power consumption. Similar to above, if justified by the application, some power consumption performance could be exchanged for a microcontroller platform that includes dedicated hardware modules that facilitate the timer device's synchronization algorithm. The trade-off between ease of implementation and possibly less measurement uncertainty with increased power consumption is one that is dependent on application.

6.2 Conclusion

This semester thesis has demonstrated the feasibility of using DCF77 as a synchronization source through the development of a timer device that is capable of generating timed events with high accuracy and low power consumption. The accuracy and availability of DCF77 was explored in detail, along with temperature dependency of microcontroller clocks. A prototype timer device with a synchronization method was designed, implemented, and tested using real DCF77 data. This semester thesis concludes that it is feasible to use DCF77 to synchronize devices, and that trade-offs between power consumption, accuracy, and implementation exist, and should be decided according to the application. Hopefully the results from this semester thesis provide useful information that may motivate further studies in this topic.

References

[1] R. Trüb, "Project Description: DCF77 Based Long-Term Timer", 2017

[2] Y. Chen, Q. Wang, M. Chang, A. Terzig, "Ultra-Low Power Time Synchronization Using Passive Radio Receivers", *"ACM 978-1-4503-0512-9/11/04"*, Apr. 2011

[3] J. I. Laveyne, L. Vandevelde, G. V. Eetvelde, "Wireless synchronisation for low cost wireless sensor networks using DCF77", *"Proceedings of the IEEE Young Researchers Symposium"*, 2014

[4] J. Sun, "Initial Presentation: DCF77 Based Long-Term Timer", 2017

[5] Wirtschaftsverlag NW, "50 years of time dissemination with DCF77", *"PTB-Mitteilungen"*, 2009

[6] conrad.ch, "DCF receiver module 641138 Suitable for series: C-Control" [Online]. Available: https://www.conrad.ch/de/dcf-empfaenger-modul-641138-passend-fuer-serie-c-control-641138.html,Dec.2017

[7] reichelt.de, "DCF77 MODULE :: DCF 77 receiver module" [Online]. Available: https://www.reichelt.de/www.reichelt.com/Kits/DCF77-MODUL/3/index.html?ACTION=3&GROUPID=7836&ARTICLE=57772,Dec.2017

[8] hkw-electronik.de, "Clock and Control Modules" [Online]. Available: https://www.hkw-elektronik.de/en/products/am-receiver-equipment/clock-and-control-modules/, Dec. 2017

[9] Texas Instruments, "MSP430FR59xx Mixed-Signal Microcontrollers" [Online]. Available: www.ti.com/lit/ds/symlink/msp430fr5969.pdf, Oct. 2012

[10] Texas Instruments, "MSP430x5xx and MSP430x6xx Family User's Guide" [Online]. Available: www.ti.com/lit/ug/slau208p/slau208p.pdf, Jun. 2008

[11] ublox, "NEO/LEA-6T u-blox 6 timing GPS modules" [Online]. Available: https://www.u-blox.com/en/product/neolea-6t, 2015

[12] L. Sigrist, "RocketLogger Datasheet" [Online]. Available: https://git.ee.ethz.ch/sigristl/rocketlogger/wikis/datasheet/, 2017

[13] Blinkenlight, "DCF77 Receiver Modules" [Online]. Available: https://blog.blinkenlight.net/experiments/dcf77/dcf77-receiver-modules/, 2013

[14] dcf77logs.de, "logs" [Online]. Available: https://www.dcf77logs.de/logs, 2017

[15] J. R. Vig, "Introduction to Quartz Frequency Standards", *"Army Research Laboratory, Electronics and Power Sources Directorate"*, Oct. 1992

[16] Atmel, "SAM L21 Family Data Sheet", SMART ARM-based Microcontrollers datasheet, Jan. 2015 [Revised Feb. 2017]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/60001477A.pdf. [Accessed Oct. 23, 2017].

[17] Microchip, "PIC16(L)F1826/27 Data Sheet", PIC16(L)F1826/27 datasheet, Jun. 2009 [Revised Apr. 2011]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/41391D.pdf. [Accessed Oct. 23, 2017].

[18] Microchip, "20-Pin USB Flash Microcontrollers with XLP Technology", PIC18(L)FXk50 datasheet, May. 2008 [Revised Apr. 2015]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/40001350F.pdf. [Accessed Oct. 23, 2017].

[19] NXP Semiconductors, "Kinetic KL02 32 KB Flash", MKL02Zxx datasheet, May. 2013 [Revised Aug. 2017]. Available: https://www.nxp.com/docs/en/data-sheet/KL02P32M48SF0.pdf. [Accessed Oct. 23, 2017].

[20] STMicroelectronics, "STM32L021", STM32L021 datasheet, Feb. 2016 [Revised Sep. 2017]. Available: http://www.st.com/content/ccc/resource/technical/document/datasheet/86/a6/5f/95/33/50/4e/d6/DM00206858.pdf/files/DM00206858.pdf/jcr:content/translations/en.DM00206858.pdf. [Accessed Oct. 23, 2017].

Appendix A

Semester Project Assignment

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Semester Thesis at the
Department of Information Technology and
Electrical Engineering

for

Jianwei Sun

DCF77 Based Long-Term Timer

Advisors: Roman Trüb

Professor: Prof. Dr. Lothar Thiele

Handout Date: 30. 10. 2017
Due Date: 05. 02. 2018

1 Introduction

Wireless IoT devices are used for numerous applications, e.g. environmental sensing, waste management, traffic congestion. It is expected that the number of wireless IoT devices will increase significantly in the next few years. Due to the large number of expected devices, maintanance which includes physical access to each device is infeasible. Therefore, IoT devices are required to have a low power consumption such that a battery lasts multiple years.

Since the available RF bandwidth is limited, it is important to use it efficiently. For most communication protocols with high channel utilization (e.g. TDMA schemes), the devices need to have clocks that are tightly synchronized. Usually this synchronization is achieved with packets sent on the data channel. The idea of this project is to use the DCF77 time signal as an alternative source to synchronize the IoT devices without wasting channel bandwidth. DCF77 is a longwave time signal transmitter located in Frankfurt which covers a large part of Europe. The advantage of this approach is the low-power consumption of the receiver (compared to a GPS receiver) and the fact that DCF77 transmits on a frequency not used by IoT devices. Since the DCF77 signal is transmitted using a very low frequency carrier, it can even be received indoors which is generally not possible with GPS. DCF77 transmits the time information in two different ways by modulating the same carrier with two different modulation schemes: amplitiude (AM) and phase (PM) modulation. The reception and demodulation of the AM signal is less complex than the reception and demodulation of the PM signal but provides less accuracy. Previous investigations on the accuracy of the DCF77 signal [1, 2] suggest that an accuracy below 1 millisecond can be achieved when using the AM component.

2 Project Goals

The goal of this thesis is to evaluate the suitability of using the DCF77 AM time signal to build a timer for long time intervals. The timer should be able to generate a digital signal at a point in time specified by a request. The device should consist of a DCF77 receiver circuit/chip and a microcontroller that processes the time signal and generates the timer events. This long-term timer device could potentially be used to generate an interrupt to wake up a microcontroller that entered sleep mode in order to save energy. The challenge is that the time difference between the request of the timer signal and the actual timer signal can be as large as 24 hours since many IoT devices sense and transmit data very infrequently. In addition, the temporal accuracy of the generated signal should be high (in the order of a few milliseconds) and the device should exhibit a low power consumption such that it can run on a single battery for multiple years.

The goals of the project can be summarized as follows:

- Investigating different options of circuits and chips to realize the DCF77 receiver and timer functionality.
- Implementation of the described long-term timer with focus on temporal accuracy, reliability and low power consumption.
- Evaluation of the obtained timer device and the suitability of using the DCF77 time signal for time synchronization.

3 Tasks

The project will be split up into multiple subtasks, as described below:

1. Study the relevant literature and research different variants of AM DCF77 receiver designs. In addition, familiarize yourself with the DCF77 signal and time synchronization.
2. Work out one ore more variants to realize a DCF77 receiver. The output of the receiver should be a digital signal which contains the time information.
3. Implement the most optimal variant of the DCF77 receiver in hardware.
4. Implement the functionality that decodes the received time signal and which takes care of handling the timer request and generating the timer events using a microcontroller.
5. Evaluate your implementation in terms of time accuracy and energy/power consumption. Furthermore, investigate the reliability of your implementation. For this, evaluate the availability of the DCF77 signal under different environmental conditions (day/night, indoor/outdoor, solar activity, etc.).
6. Compare the performance of your implementation with other timer implementations in terms of energy consumption, temporal accuracy and reliability.
7. Document your work in a written report.

4 Project Organization

4.1 Weekly Meeting

There will be a weekly meeting to discuss the project's progress based on a schedule defined at the beginning of the project. A revision of the working document should be provided at least 24 hours before the meeting.

4.2 Deliverables

- **Time Table:** In the beginning of the project, a time schedule is developed. The time schedule includes milestones of the project.
- **Initial Presentation:** In the first month of the project, the topic of the project will be presented in a short presentation during the group meeting of the Computer Engineering Lab. The duration of the talk is limited to 5 minutes.
- **Final Presentation:** At the end of the project, the outcome of the project will be presented in a 15 minutes talk, again during the group meeting of the Computer Engineering Lab.
- **Report:** All aspects of the work are to be documented in a written report. The report should discuss the context, design, implementation and evaluation of the work. Furthermore, the report should include the assignment and the time schedule. A hard-copy of the report is to be turned in.

- **Software:** A copy of the developed software in digital form has to be handed in at the end of the project.

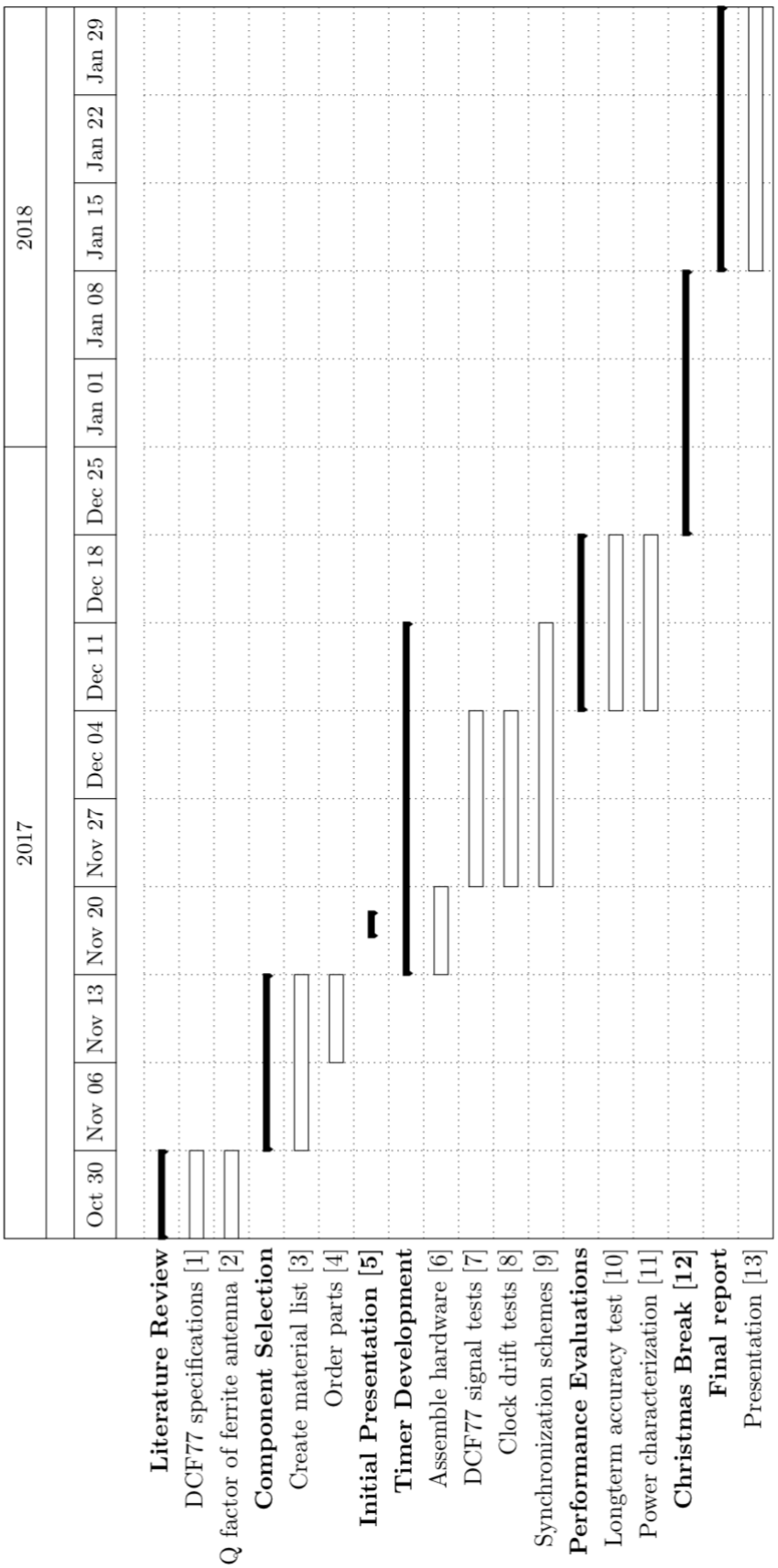
References

[1] Y. Chen, Q. Wang, M. Chang, and A. Terzis. Ultra-low power time synchronization using passive radio receivers. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 235–245. IEEE, 2011.

[2] J. Laveyne, G. Van Eetvelde, and L. Vandevelde. Wireless synchronisation for low cost wireless sensor networks using DCF77. In *Young Researchers Symposium 2014*. EESA, 2014.

Appendix B

Project Timeline


2017
2018
Oct 30
Nov 06
Nov 13
Nov 20
Nov 27
Dec 04
Dec 11
Dec 18
Dec 25
Jan 01
Jan 08
Jan 15
Jan 22
Jan 29
Literature Review
DCF77 specifications [1]
Q factor of ferrite antenna [2]
Component Selection
Create material list [3]
Order parts [4]
Initial Presentation [5]
Timer Development
Assemble hardware [6]
DCF77 signal tests [7]
Clock drift tests [8]
Synchronization schemes [9]
Performance Evaluations
Longterm accuracy test [10]
Power characterization [11]
Christmas Break [12]
Final report
Presentation [13]

[1] Reviewing specifications include understanding the analog specifications of the signal as well as the digital message encoded. For instance, being able to decode the time from a complete DCF77 transmission.

[2] It is necessary to understand why a ferrite antenna is used, and how a closely placed capacitor creates an LC tank that filters the designed carrier frequency.

[3] Having selected a microcontroller, prepare a list of the necessary components required for microcontroller function. For example, if a development board is available, it would help with initial firmware design. Otherwise, discrete components are needed to assemble the system. Component selection also includes the necessary DCF receiver modules.

[4] It would be most convenient to source all components from one supplier, i.e. DigiKey. Assuming products are in stock, shipping from sources such as DigiKey or Conrad takes approximately one week.

[5] Initial 5 minute presentation on topic introduction, motivation, project goals, and planned strategy. Will occur at 9 AM on 23.11.2017 in the meeting room.

[6] Once ordered components have arrived, they will be assembled into a prototype that can be used for initial testing. The prototype should be customizable and debuggable to allow for any design modifications if necessary.

[7] Studying the DCF77 signal using the prototype will provide information on signal availability and reliability. This information will motivate the design of the synchronization scheme.

[8] The Timer Device can be put in a number of different environments (hot outdoors, cold indoors, etc...) to measure how its local time drifts after a certain duration of time has elapsed. These results will provide an idea for the periodicity of synchronization required to achieve the desired accuracy, and motivate the design of the synchronization scheme.

[9] The synchronization algorithm will govern the appropriate sleep state transitions of microcontroller to save power, while performing the required time calibration calculations. This algorithm will be designed to dynamically adjust its synchronization periodicity to optimize for power consumption and time-keeping accuracy.

[10] The accuracy testbench will instruct the Timer Device to trigger events at certain intervals (mins - hours), while measuring the elapsed time between each event. Measured time data will then be compared to the expected trigger event times to gauge the overall accuracy and precision of the Timer Device.

[11] The power testbench will be able to report voltage and current information for the Timer Device. Power consumption in active and sleep regions can be characterized separately as long as the switching between these two starts is properly understood.

[12] During the Christmas break, access to equipment may be limited. As a result, data collection should be completed before this time. Result analysis can be performed during this time.

[13] Once all data has been gathered and processed, it will be documented in the final report. A presentation of the entire semester project will be delivered upon completion.