

# Alarm Clock Addon

## Overview

This [addon](#) extends a [Shrimp Parts](#) kit to create a functioning digital clock. The build offers a starting point for all kinds of time-driven inventions.

The [project build](#) shows how to wire, program and configure the circuit as a clock which chimes hourly, or which sounds according to a daily or yearly schedule.

For a visual display of the time, the Alarm Clock kit can be augmented with an [LED Clock Addon](#) which incorporates 24 superbright LEDs and high-power LED driver chips for a Unary clock or Word Clock.

A set of [activities](#) are provided for educators who wish to use these builds to support lessons in high school or university.

## Design

The build employs a Real Time Clock (RTC) chip which has a precise quartz-controlled time signal, as well as a battery-backup system, so that the time doesn't need to be reset every time the clock is unplugged.

Since the RTC chip keeps time independently, the [Shrimp](#) can be placed in a very [low-power mode](#) suitable for long-lived projects with time-driven behaviours running from battery.

## Learning Outcomes

This project shows how to use peripheral Integrated Circuits (ICs) as part of a @ShrimpingIt layout, demonstrates how to use a number of distinct digital communication protocols, and some special IC configurations. \* Protocols - Serial without clock - used to set and read the time (CP2102 UART IC) - I2C - used to set the clock, configure it, and read the time when needed (DS1307 Real Time Clock IC) - Serial with clock - controls the LEDs (WS2803 LED Driver IC) \* Configurations - The Shrimp configures itself to go into a low-powered sleep mode - The RTC is configured over I2C to flash an LED once per second

Many D&T and Computing tasks are possible as spinoffs, specialising the housing, the time-controlled outputs or the software behaviour to suit many different purposes limited only by the imagination.

## Discussion

Learners may remember from the original [Blink build](#), that the ATMEGA328 requires a 16MHz crystal component, as part of its internal 'clock' circuit. Hz

is short for Hertz, a scientific unit meaning ‘per second’, so a 16MHz crystal provides a ‘tick’ 16 million times a second.

The ATMEGA’s clock circuit is used to trigger computing instructions. Each instruction relies on the results of the one before, and the 16MHz tick gives enough time for the movements of electrons which make up a single computing instruction to be completed before the next is triggered.

Although it is accurate enough to space out computing instructions, the 16 MHz signal is not very precise in absolute terms. If we were to rely on it to control our digital clock, adding up the inaccuracies over a whole day, we’d end up losing or gaining time making the clock unreliable. Even if we were to sample and assess the drift of our 16MHz crystal, and attempt to compensate for it, a later change in room temperature would influence the speed of the tick!

The Real Time Clock (RTC) Chip is designed to provide a much more precise timing circuit, stable to ambient temperature changes, for use in digital clocks.

In this build, we use the RTC as a peripheral chip, which is initially set by a user sending the correct time over the Serial UART to Arduino code running on the ATMEGA328. Arduino code running on the Shrimp then sets the RTC by sending a signal over the I2C bus. Once set, the RTC is then consulted periodically using an I2C request to establish an accurate time signal which can be used in our application.

Even when the rest of the circuit is shut down, the coin battery can keep the Real Time Clock chip active. When power comes back to the rest of the circuit, the time reported by the RTC will still be accurate.