OFFICIAL JOURNAL OF THE BRITISH HOROLOGICAL INSTITUTE

The Horological Journal

MAY 2022
www.bhi.co.uk

Remote Monitoring and Control of a Turret Clock

An 1826 Turret Clock Meets the Internet of Things

Martin Rice

Nowadays the Internet not only connects people across the world together, but so-called 'smart' devices such as your smart watch, your central heating, maybe even your fridge, as well. The term 'Internet of Things' (IoT) refers to this network and electronics companies have developed inexpensive devices to allow 'things' to connect to it. This article describes how such devices have been used to monitor the hourly behaviour of the turret clock in the church of St Nicolas, Newbury, and to upload the information to a page on the World-wide web.

Several years ago I took responsibility for the week-by-week checking of the church clock. The clock often gained or lost perhaps five minutes over the week, and adjusting the pendulum bob up and down didn't seem to have the expected effect.

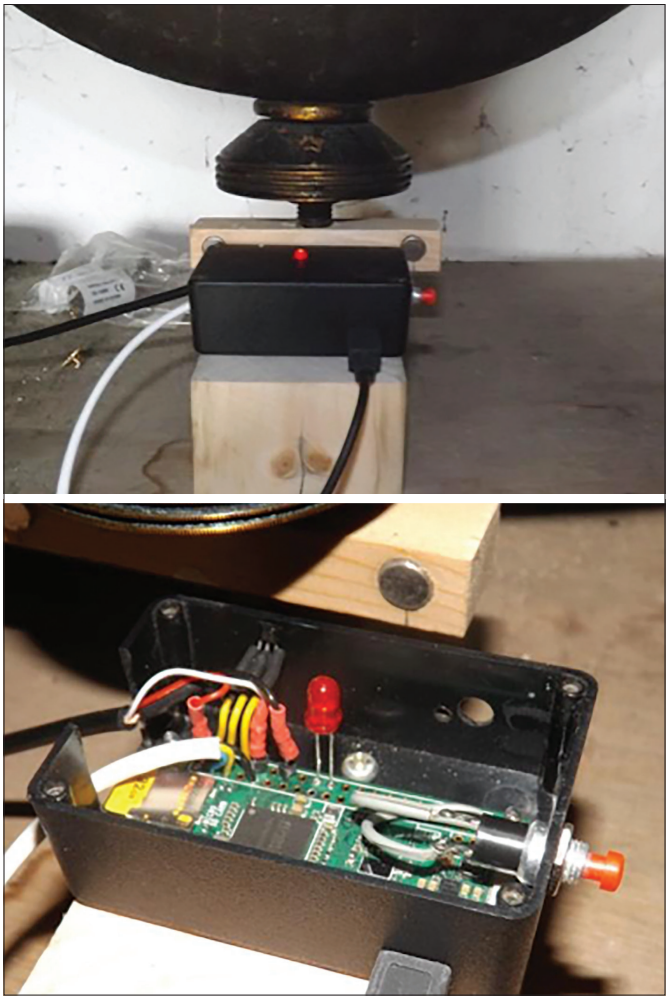
As a lecturer in electronics engineering, I thought it would make a good HNC project for someone in my class to design and build a microprocessor system to monitor the clock behaviour by timing the swings of the pendulum. I wrote a design specification, but unfortunately no-one took up the challenge, so I decided to do it myself. A few months later a low-power laser and phototransistor was detecting each swing of the pendulum, and a black box of electronics, with lots of switches and push buttons and a small LCD display, was recording a week's worth of data.

It confirmed that the clock was behaving erratically. The first breakthrough in getting the clock to keep better time was to remove some of the weight from the drive system. The escapement was being over-driven, striking its bankings rather than swinging cleanly. The second discovery was that the clutch, which was part of the leading-off work, was slipping.

Correcting these two issues led to a much better-behaved clock, with a period of exactly three seconds (I use the term 'period' as a physicist would: the time taken for a complete to-and-fro cycle of the pendulum). Tweaking the pendulum bob, or adding/removing small auxiliary weights to a platform attached to the pendulum rod, allowed me to control the clock so that it kept time to within a few seconds per week.

This system worked well for many years. However, when I retired from lecturing, I wondered about updating the electronics. I had seen one of my more able HNC students using a little computer called a Raspberry Pi to do amazing things. The idea of getting a Raspberry Pi to measure the pendulum swings and to upload the information to a web page grew in my mind and became an absorbing project.

With the help of a colleague, and the many resources on the Internet, I learned how to programme Raspberry Pi computers. With my electronics background I was able to interface the pendulum swing detector to the little computer and, via the church wi-fi, uploaded hourly timekeeping information to an IoT page on the ThingSpeak website.



Figures 1A and 1B. The black box hardware.

Figures 1A and 1B show the hardware. The black box contains the tiny Raspberry Pi computer, an LED and a push switch. The switch is used to set things up and the LED helps in this process. The black cable going off to the left connects to the hour strike sensor (more detail later), and the white cable goes to a temperature/humidity/air pressure sensor. The other black cable supplies power from a standard 5V USB phone charger.

Instead of using a laser diode and phototransistor to detect the swings of the pendulum, this version used magnets and a

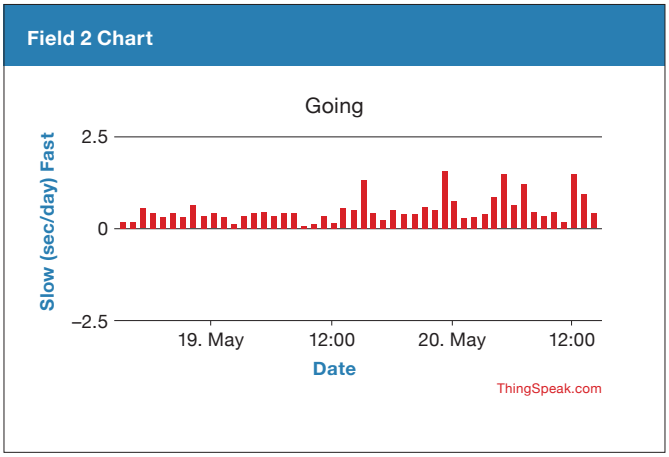


Figure 2. Rate ('going'), measured by Raspberry Pi.

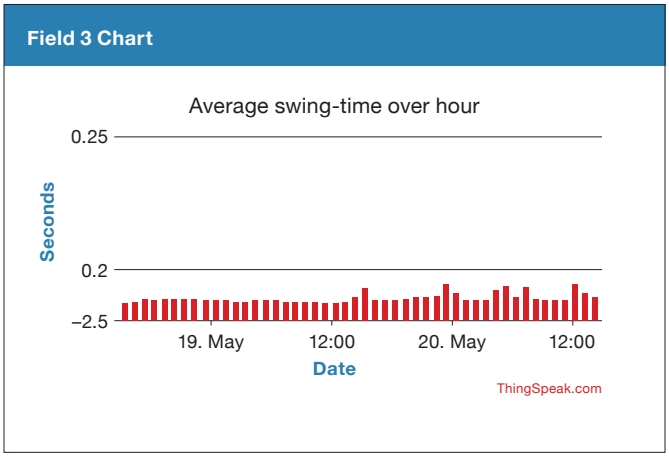


Figure 3. Average swing-time measured over an hour.

solid state sensor called a Hall-effect switch. The sensor is the thing with three yellow legs inside the black box.

The magnets can be seen in the photo, and are housed in a block of wood fitted to the very bottom of the pendulum. Having two magnets arranged as shown allows the system to measure the linear speed of the pendulum as it swings past bottom dead centre, and hence obtain a measure of the angular amplitude through which the pendulum is swinging.

The amplitude is not actually calculated, but the swing-time (the time taken between one magnet passing the senor and the other) is recorded and displayed. With the magnets 80mm apart the swing-time is typically 0.2 seconds.

As well as recording swing-time, the computer counts complete ('tick' and 'tock') swings of the pendulum, and when this reaches 1200 it measures how long it took. With a three-second period, this should take one hour, of course. Time is measured to the nearest millisecond, and any error in the rate is converted from milliseconds per hour into seconds per day.

Once an hour, the Raspberry Pi uploads the information to a page on the ThingSpeak website. The graph, **Figure 2**, shows the rate ('going' error) for a couple days in May 2020. It shows that the clock, on average, is gaining very slightly, although by less than a second per day: quite an acceptable situation. The occasional spikes in the data are puzzling, and I will come back to that later.

Figure 3 shows the corresponding swing-time graph. This data is closely (but inversely) related to the amplitude of the swing. Comparing **Figure 2** with **Figure 3** shows a close correlation between the amplitude of the swing and the rate. This is a manifestation of circular deviation.

The clock might be running at the right rate, but is it on time? A second sensor is required to detect exactly when the clock strikes the hour. **Figure 4** is a photo of the strike sensor. The black box contains another Hall-effect switch, and the magnet can just be seen on the tail of the hammer lifting lever. The results are shown in **Figure 5**.

Figure 5 seems to show that the hour strike occurs somewhat randomly, even though the pendulum is keeping good time. For instance, near the centre of the chart the 12 o'clock strike on 19 May is about two seconds fast (early); at 1pm it was 2.5 seconds early, then at 2pm it was two seconds late. These variations are clearly not caused by changes to the pendulum motion, but there must be some feature of the strike work that creates them.

Measuring temperature, humidity and air pressure is fairly

Figure 4. Hour strike sensor and magnet.

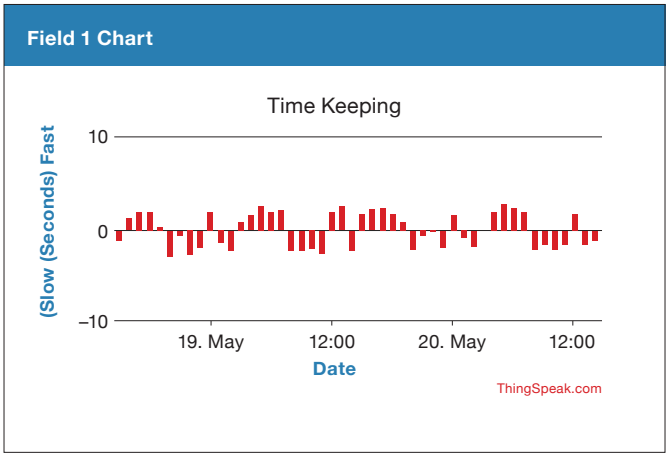
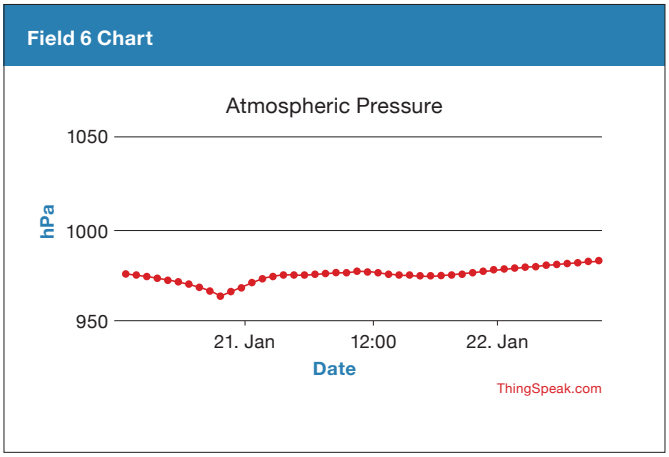
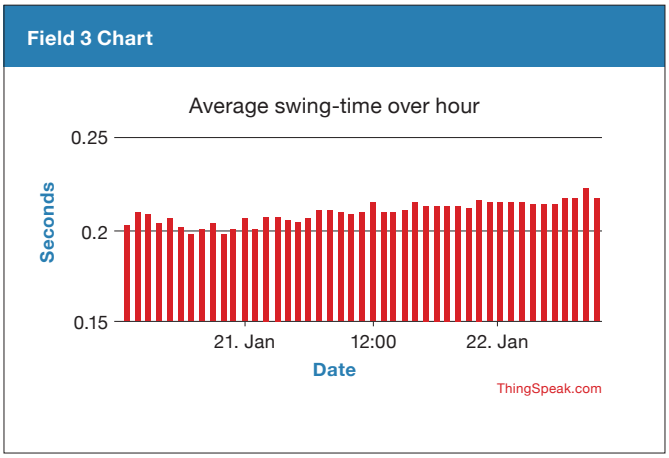


Figure 5. Time of the first strike of the hour, measured by Raspberry Pi.

straightforward using standard electronic sensors. I thought it would be interesting to see what correlation there was (if any) between these parameters and the behaviour of the clock. I was rather disappointed to find that my favourite theories about why the clock gained or lost time were not borne out. One set of data was quite interesting, however; when Storm Christoph went through Newbury in January 2021, **Figure 6**.





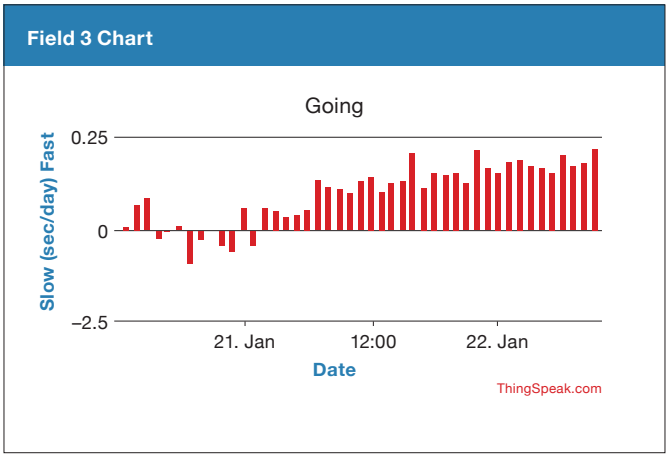


Figure 6. The effects of Storm Christoph, 21–22 January 2021: air pressure, average swing-time and rate.

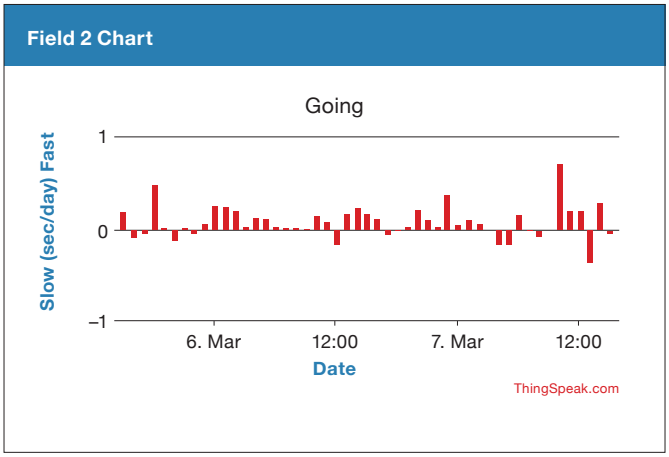


Figure 7. Rate measured by Raspberry Pi.

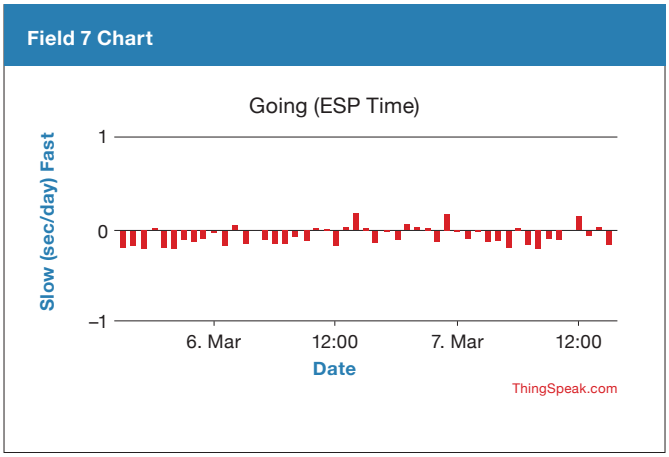


Figure 8. Rate measured by ESP32.

Anomalous Behaviour of the Raspberry Pi

The Raspberry Pi is an astonishing piece of technology, but I had my doubts as to the reliability of the information it was giving.

I set up an experiment to measure the rate using a Raspberry Pi and another device referred to as an ESP32. The ESP32 is a microcontroller, even smaller than the Raspberry Pi, and which also contains on-board wi-fi and bluetooth capable of connecting to the Internet directly. **Figures 7 and 8** show the results: a bit of an embarrassment for the Pi!

The reasons for the difference in this context as that the Raspberry Pi automatically updates its timekeeping to keep in step with 'Internet time (NTP)'. This is a matter of microseconds every few minutes, enough to interfere with the measurement of the clock rate. The ESP32 doesn't do this. If left to its own timekeeping it would slowly drift, but I took measures in the software, and initial calibration, to minimise this. The Raspberry Pi operating system makes it inappropriate for measuring time intervals over an hour to millisecond accuracy. The ESP32 has different problems but these are easier to overcome. These involve initial calibration, and re-synchronising with NTP, once an hour, under control from the software I wrote.

The measurements obtained from the ESP32 are really quite impressive. It provides a resolution of just one millisecond over one hour so the required accuracy is one part in 3,600,000, or about 0.3 parts per million. The software calibration factor brought into the calculations as mentioned above is typically 0.999996. Every ESP32 used has to be individually calibrated and I use a GPS module to do this.

Like the Raspberry Pi, the ESP32 has wi-fi capability, plenty of programming power, has lots of pins to interface to sensors, costs under £5, and is now the device I use for monitoring the clock at Newbury.

Connecting to the Internet Using GPRS

Having wi-fi in the church at St Nicolas is unusual. For more general situations it would be necessary to communicate with the ThingSpeak website without wi-fi. The old (2G) mobile phone network uses GPRS (General Packet Radio Service) to

exchange data, and it has been possible to incorporate this into a clock monitoring system.

Figure 9 shows a combined ESP32 with a SIM800L module. This allows connection to the GPRS cellular (phone) network. The antenna is the black strip that looks a bit like a length of insulating tape. On the underside of the board is a SIM card. The container is a single-gang 25 mm deep electrical pattress, and the two round connectors are for Hall-effect sensors for the pendulum and the hour strike. Not all phone networks support 2G, but Giffgaff does, and it costs less than a penny a day (on pay as you go) to upload the data every hour.

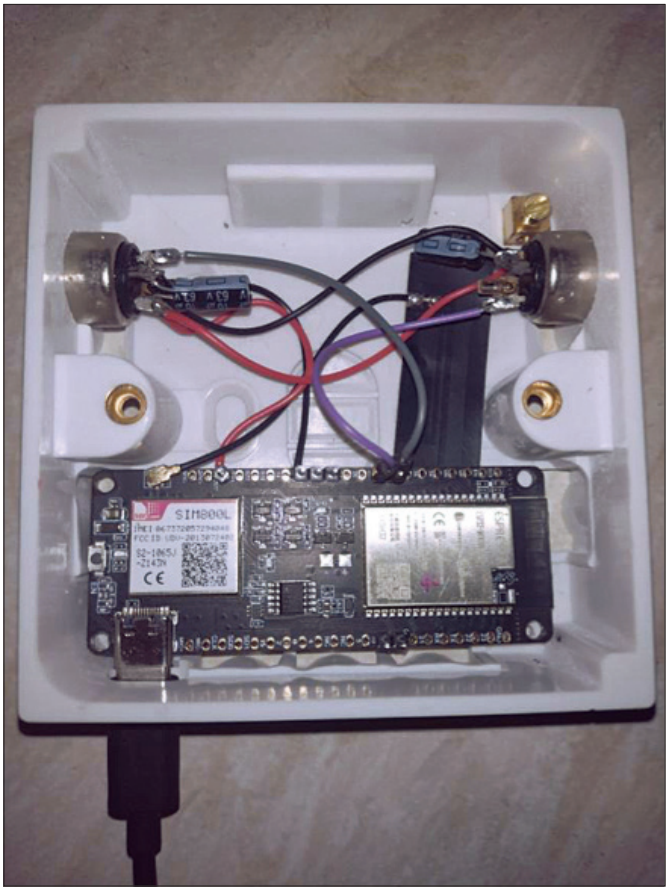


Figure 9. ESP32 with SIM module.

Remote Control of the Clock

It is a common practice to adjust the rate of a turret clock by adding or removing small weights from a platform attached to the pendulum rod. The approach I have used is not to add or remove weights, which would be mechanically quite tricky, but to move a given weight up the rod a few millimetres (to speed things up), or to lower it (to slow things down).

Two pieces of wood, **Figure 10**, have been clamped to the pendulum rod about 200 mm apart, the lower one just above the pendulum bob. Spanning these are two smooth vertical metal rods. These serve as guides for a small stepper motor. The motor has a threaded rod coupled to its output shaft and passes through a threaded bush in the upper piece of wood. As the stepper motor shaft rotates, the motor winds itself up or down. A further plate is mounted to the motor assembly containing some pieces of lead.

At Newbury the total weight of the motor assembly is about 500 gm and the main bob is estimated to be about 100 kg. Turning the motor shaft through ten complete turns (raising or lowering the auxiliary weight by 8 mm) causes the rate to change by about half a second per day.

Figure 10. Stepper motor and auxiliary weight assembly.

A stepper motor was chosen for this application because it is easy to control. My initial experiments used a Raspberry Pi computer, with an interface to drive the stepper motor. With this set-up I could monitor the behaviour of the clock from my PC at home and, using Virtual Network Computing, I could activate the stepper motor and tweak the rate as the need arose, saving the trouble of having to go along to the tower to make manual adjustments.

When wi-fi communications were discarded in favour of GPRS, the stepper motor module was incorporated into the ESP32/SIM system, and it is now possible to control the clock by sending text messages.

Automatic Control

The ESP32 microcontroller knows whether the clock is fast or slow, and it knows if it is gaining or losing time; it can also control the rate by altering the position of the auxiliary weight. It should be possible to write an algorithm that drives the stepper motor to keep the clock to time without the need for human intervention.

The system I have implemented at the moment monitors the timekeeping and rate for 12 hours, from midnight to midday. At midday it drives the stepper motor to adjust the rate so that

Figure 11. Data compiled with ThingSpeak showing automatic control of the clock.

the clock will be on time twelve hours later (next midnight). At midnight it returns the stepper motor to the position that gives nil gaining or losing and re-enters the monitoring regime.

Figure 11, shows the system behaving more or less as expected. From midnight to midday of 4 September the timekeeping was a little slow – by about 1.33 seconds, on average. At midday the system automatically drove the stepper motor from position 42 up to position 104, and consequently the rate error jumped up to about three seconds per day fast. By midnight on 4 September the clock had gained about 1.5 seconds, and so the timekeeping was no longer slow. In fact it seems to have slightly over-compensated and made the clock run a little fast. At midnight the stepper motor was driven back down to position 46 and monitoring mode resumed.

One limitation of this system is that the stepper motor is connected to the microprocessor by a four-core cable. The cable is quite light and flexible, and has been routed from the control box up to the top of the clock. It then follows a large unrestricted arc to curve over before being attached to the top of the pendulum rod and clipped on its route down to the assembly. The effect of this on the behaviour of the pendulum has not been quantified. A similar unknown effect is how the less-than-aerodynamic shape of the assembly itself will affect the pendulum behaviour. Further research into these issues might lead to the development of a system in which the stepper motor is battery driven and controlled wirelessly by its own on-board ESP32. The batteries could form part of the added mass currently provided by the lead. I have since installed similar systems in a clock somewhat older than that at Newbury, as well as on an 1882 Gillet and Bland clock with gravity escapement.

Conclusion

For the price of about £20 a system has been developed that tells you hour-by-hour what your church clock is doing, and allows you to view the information from anywhere connected to the Internet. In the current system, time-keeping and temperature, humidity and atmospheric pressure are uploaded via GPRS and the mobile phone network. The system is self-regulating, but you can control the behaviour by sending SMS messages to the clock. No irreversible alterations have been made to the clock itself, in line with the *Code of Practice for Turret Clock Work* issued by the Clock Advisers' Forum.

For live observation of the system go to https://thingspeak.com/channels/791978

I would be happy to give more details to anyone interested. Please contact me at jmartinrice@gmail.com.

A Final Note from the Author…

As well as the turret clock at St Nicolas Church, Newbury, I am now monitoring my longcase clock as well as three turret clocks. The ThingSpeak pages can be found at:

https://thingspeak.com/channels/1409501
https://thingspeak.com/channels/1289670
https://thingspeak.com/channels/983612