

# ARDUINO ESP8266 BASED SPEAKING CLOCK

Author: 6V6GT

Version 1.00 22-July-2018

## 1 INTRODUCTION

This is an Arduino audio project designed to replicate the functionality of telephone network speaking clock services and includes some historical English and German language voices that were in operation for telephone subscribers in the past. For example, that of Pat Simmons from England active (1963 - 1985) and Renate Fuczik from Austria active (1984 - 2009). The user can create further files as required.

The basic hardware design is quite simple. The main components are an ESP8266 with the flash memory holding a pair of voice and index files and the logic for building up the spoken text and an I2S DAC for producing the audio output. I've added an amplifier and loud speaker and a connection for a standalone telephone, but these are optional. This device can NOT be connected to a telephone network and, apart from the regulatory considerations, will not tolerate the around 90 volt AC ring voltage on such networks.

The device is configured via a web browser by joining it to an existing wlan or it can create its own wlan if required (AP Mode). The main time source is NTP (network time protocol) and you can add an optional real time clock which actually is recommended for a quicker startup or if there is no internet connected wlan available. Of course, you can integrate other time sources if you wish such as a DCF77 radio time module or a GPS module but then you'll have to modify the software yourself.

The player is a software component which uses a voice file (actually a .wav file) containing all the phrases and numbers concatenated together and with an index file. There is a "speech engine" which uses a workflow to drive the player, instructing it which phrase or number it should play at what time in the cycle. New voice / index file pairs can be loaded into the ESP8266 flash memory using a built-in utility.

The project has some interesting design features which could be useful for other applications, for example the NTP clock part is non-blocking (unlike most designs which issue an NTP request and block while waiting for the response) and it handles fractions of a second, scheduling a system time update for the next full second to maintain a sub second degree of accuracy. Also, the browser interface has the feature that it appears as multiple pages, but is in fact a single HTML page where all but the current logical page are hidden. This allows the user to navigate between pages without causing intervening returns to the server.

The project could be extended or adapted for other "announcement like" purposes where audio phrases are built up out of separate words depending on some criteria e.g. time, measurements from an instrument, say temperature or other such like weather information, etc. etc. and may be useful for people with impaired vision.

There is at least one other similar project in existence. The Telecommunications Heritage Group have a design [Ref:4] (PIC assembler based) which appears very professional and full featured and can even (subject to regulatory approval) be connected to a telephone network to provide a speaking clock service. However, it uses an SD card player with pre-built phrases as opposed to the alternative used in the design presented here of complete dynamic assembling of the announcement text. Further, the system design is closed in that the software source code is not published. To support their device, they have collected a number of historical voice files from various sources.

The copyright situation regarding the voice files extracted from the various commercial devices may not in all cases permit unrestricted commercial usage so you are advised to check this for yourself in the case that you intend to use any of such files for some commercial purpose.

This is not an ideal project for an absolute beginner and a basic familiarity with the Arduino development environment for the ESP8266 is a prerequisite as is the ability to build hardware from a schematic design and manufacturer data sheets.



Speaking Clock connected to a vintage phone.

## 2 CONTENTS

1	Introduction.....	1
2.1	Acknowledgements.....	4
2.2	Abbreviations .....	4
3	Solution Description.....	5
3.1	Overview.....	5
3.2	Function Block Diagram.....	5
3.3	Main Software Module Description.....	6
3.3.1	Time Handling .....	6
3.3.2	Configuration Management.....	6
3.3.3	Player Support and Spiffs file system .....	7
3.3.4	Display Manager .....	7
3.3.5	Application Control Logic .....	7
3.4	Schematic Diagram.....	7
3.5	Main Components/ Modules Used.....	9
4	Construction .....	11
5	User Manual .....	12
5.1	Initial set up.....	12
5.2	Web Configuration.....	13
5.3	LCD screen .....	16
5.4	Control Button.....	17
5.4.1	Toggle amplifier on/off.....	17
5.4.2	Display configuration information on the LCD screen.....	17
5.4.3	Force device into AP Mode .....	17
5.4.4	Force Factory Reset.....	17
5.5	File formats.....	18
5.5.1	Wave (.wav) file .....	18
5.5.2	Track (.trk) File .....	18
5.6	Creating voice recordings.....	19
5.7	Creating Workflows .....	20
5.8	Error Numbers.....	20
6	Possible Future Development .....	21
7	trouble shooting tips.....	21
8	Software Distribution .....	21
9	Appendix .....	22

## 2.1 ACKNOWLEDGEMENTS

- Arduino Library Creators not mentioned elsewhere in the text: Ivan Grokhotkov, Earle F. Philhower and many others.
- Telecommunications Heritage Group (see introduction)

## 2.2 ABBREVIATIONS

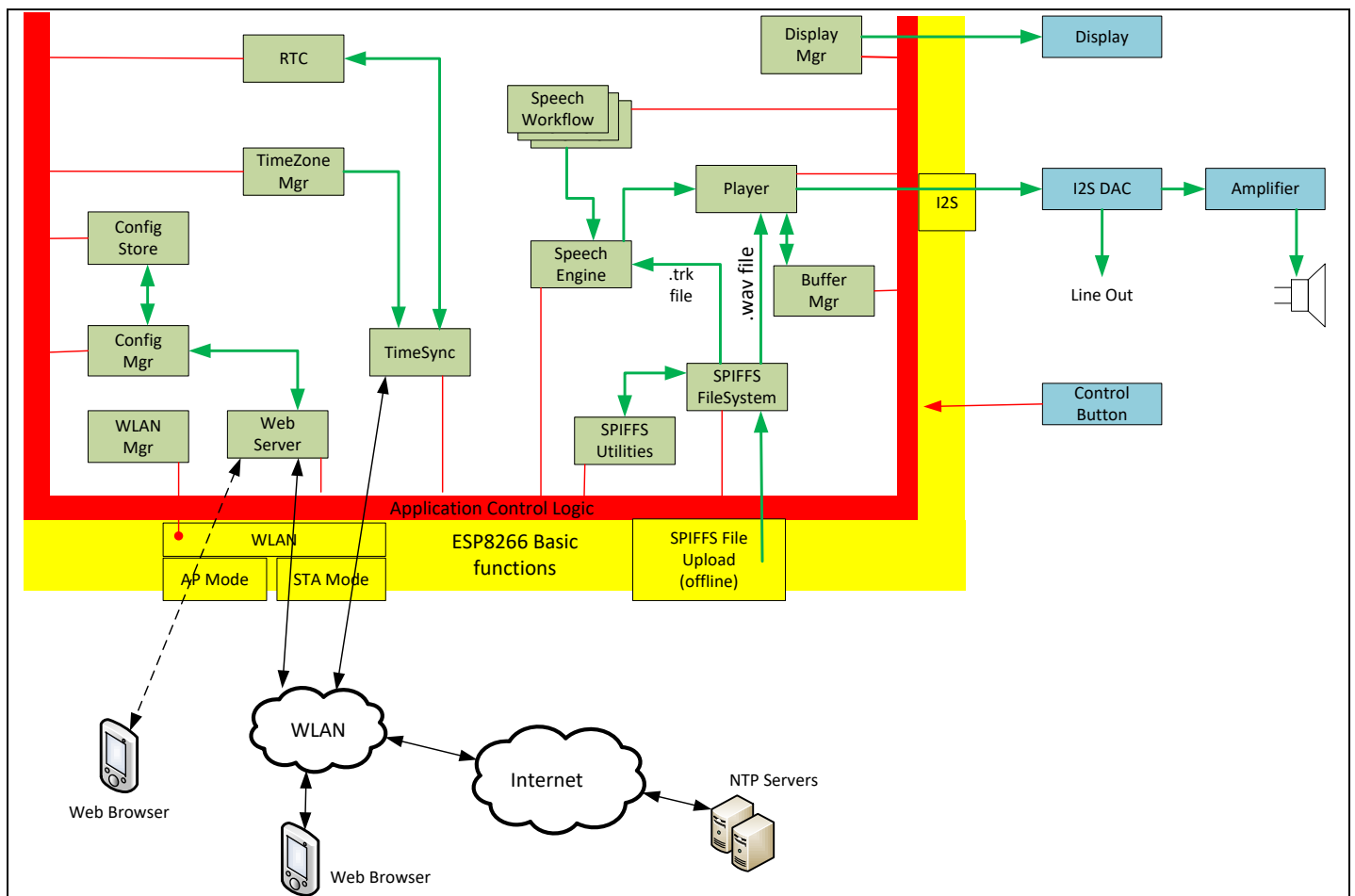
ADC	Analog to digital converter
AP Mode	Access point mode. In this mode, the ESP8266 creates an access point which wireless devices can connect to. It is used for an initial configuration or where no other WLAN is available.
Audacity	An audio file editing/conversion program
DAC	Digital to Analog converter
DST	Daylight saving time.
Eeprom	Electrically erasable read only memory
ESP8266	A microcomputer compatible with the Arduino development environment.
Flash Memory	Rewriteable permanent storage
I2S	An digital audio data transmission protocol
IC	Integrated Circuit
IDE	Interactive Development Environment.
.ino File	The main file of an Arduino program.
IP Address	Network address of a connected device in example format 192.168.1.99
Line Output	A standard level of audio signal suitable for direct input into an main amplifier.
NTP	Network time protocol
PCM	Pulse Code Modulation - an audio file encoding format.
PSK	Pre-shared key - a WLAN equivalent of a password.
RTC	Real time clock - a crystal controlled time keeping module.
SMD	Surface mounted soldered on components (no connecting wires)
SPIFFS	File system implementation in flash memory
SSID	WLAN network name
STA mode	An ESP8266 where the device is used as a wireless client of an existing network
UTC	A standard time: Coordinated Universal Time similar to GMT
WAV	Audio file format
WLAN	Wireless Local Area Network

## 3 SOLUTION DESCRIPTION

### 3.1 OVERVIEW

An ESP8266 based microcontroller using the Arduino core software drives the system. It has a built in wireless lan controller which is used for both system configuration through a web browser and for synchronization of the system time via an NTP server. An optional RTC module is also maintained from the NTP and used as a backup secondary time source for when NTP is not available e.g. at system start. A pair of files, a wav file and an index (.trk) file is used to build the audio phrases for time announcement and these are stored in the flash memory of the microcontroller. The wav file must be encoded as PCM 16bit with a 16kHz sample rate. The .trk file is an index into the wav file so each number or phrase can be referenced. A speech engine assembles these audio phrase parts according to a time based work flow with a 10 second cycle. The digitally encoded audio signal is transmitted through an I2S bus to the DAC module which decodes the audio the signal to line output level. The specified DAC module can, with a series resistor and capacitor, drive a standard telephone earpiece. An optional amplifier and speaker are specified and can be switched on for a configurable time period on pressing the control button.

### 3.2 FUNCTION BLOCK DIAGRAM



### 3.3 MAIN SOFTWARE MODULE DESCRIPTION

There is a near 1:1 relationship between the software modules identified in the diagram above and the C++ compilation units in the program.

For further details, see the source code and comments.

#### 3.3.1 *TIME HANDLING*

The internal time of all devices is maintained in UTC. At the point where it is used (display / audio) it is converted using offset for the selected timezone. Similarly for manual time entry, if used. The time is entered by the user in local time and converted to UTC. Sub second accuracy is achieved for both NTP and RTC time sources. The RTC is optional and ignored if not present.

##### 3.3.1.1 TimeSync

This obtains a time from NTP (where available) and maintains the System time and the RTC. The system time and RTC have a granularity of 1 second. NTP delivers a time including fractions of a second so time updates are scheduled for the next full second. Priority of the time sources is enforced here. Only if NTP is or becomes unavailable, typically at system start, does the RTC get used as a time source.

##### 3.3.1.2 TimeZoneManager

This is based on a design by Jack Christensen with a small modification to permit dynamic updating instead of being the timezone being defined only at system start.

##### 3.3.1.3 RTC

This is also a modification of a standard design intended for a predecessor of the DS3231 chip with corrections to make it compatible with the ESP8266. A future activity is select a completely standard library (there are many to evaluate) and integrate it here. Special is the configuration feature to generate a 1Hz squarewave where the falling edge indicates a seconds rollover.

#### 3.3.2 *CONFIGURATION MANAGEMENT*

When the ESP8266 is first loaded with the Speaking Clock software, it has a factory configuration. The configuration is managed through a web server which the user contacts through a web browser. A WLAN manager can create a wireless access point (AP Mode) or, if credentials are available, join an existing WLAN.

The configuration is stored in flash memory (emulated Eeprom) so it is preserved with the device is powered off, and a copy is maintained in RAM from the time the device is switched on.

##### 3.3.2.1 WLAN Manager

This selects which wireless mode is active for the device. Normal is STAmode where the device has the access data to join an existing WLAN. An non-configured device uses AP mode where the device creates its own WLAN.

The control button can also be used to force a device into AP mode if required.

##### 3.3.2.2 Web Server

This is configured to deliver a static web page to the browser containing all the logical pages for the configuration settings. When the browser loads this page, a return to the web server is initiated to collect all the dynamic data in the form of a JSON string. If the user makes changes in the browser and hits the "update" button, these changes are returned to the web server and loaded back into flash memory.

##### 3.3.2.3 Config Manager

This is used to load the flash memory configuration into the Config Store in RAM and, in the case of changes, copies the RAM version back into flash memory so that at the next startup of the device, the newest changes will still be available. If the device has never been configured it will not have a special marker (referred to as a

magic string in the program code) in the flash memory so the factory configuration is installed. Further, if the format of the flash memory has been changed, for example to add new configuration parameters, then the magic string must also be changed to force the flash memory data back to a factory condition.

### *3.3.3 PLAYER SUPPORT AND SPIFFS FILE SYSTEM*

The (currently 3MB) file system holds a matched pair of .wav and .trk files. The files are loaded using the ESP8266 Sketch Data Upload Tool. At system start, the .trk file is validated and loaded into RAM. The .wav file is located and opened and the header part is interpreted to ensure the encoding is correct for this device. The .wav file remains open.

#### **3.3.3.1 Speech Engine**

This performs the validation and loading of the .trk file at system start. During operation it runs the workflow defined in the .trk file at 10 second intervals. The workflow identifies the phrase or number that should be played at a specific instances during the cycle and passes these to the player, that is it passes the start time, end time and start instruction.

#### **3.3.3.2 Player**

The player, on receiving an instruction to play, fetches the data from the .wav file using a seek / read method and maintains a group of 4 buffers in a circular configuration. A timer calls a function every few mS to extract data from the buffer , process it by converting the contents for I2S sending these on to the I2S subsystem. When the end of the audio data is reached, the buffers are continually filled with audio silence (actually a zero) for 16bit encoded .wav files. Should you wish to experiment with other encoding formats (and the DAC can handle these), then you make changes in the Player module.

#### **3.3.3.3 Spiffs Utilities**

This includes the code, based on work by bbx10 [Ref: 3] for interpreting .wav headers.

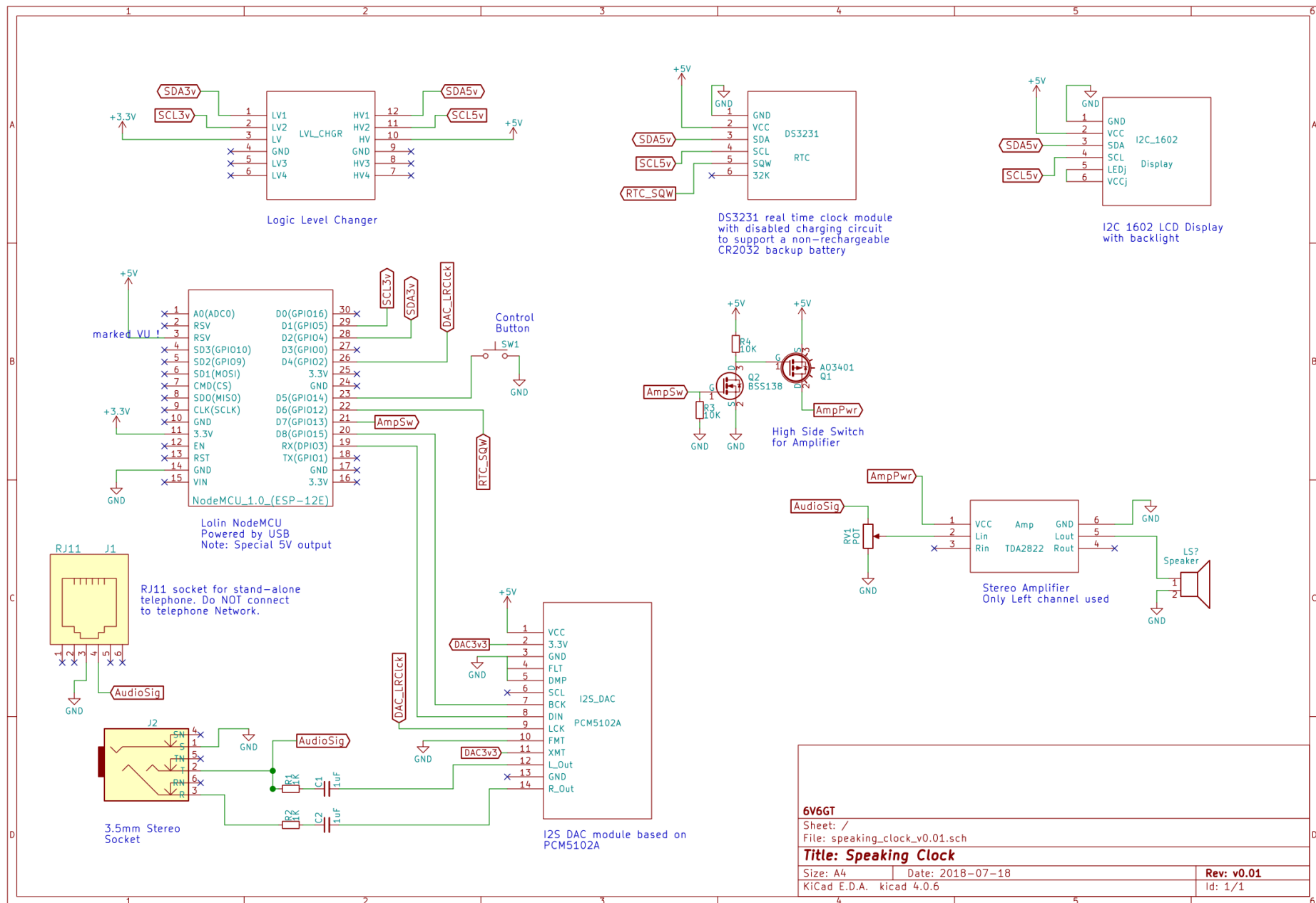
### *3.3.4 DISPLAY MANAGER*

This is relatively simple because the chosen LCD screen (with I2C backback) is very easy to drive. This contains the definition of the special symbols (see user manual), the formatting of the time and date in the display and the logic for the timed sequences of multiple pages e.g. configuration information.

### *3.3.5 APPLICATION CONTROL LOGIC*

The .ino file has a setup section and a loop section. The setup section starts, in a controlled order, the setup routines for all the software modules. Similarly, the loop section, on each iteration, calls the loop section in each of the software modules. The handling of the control button is also done here.

## **3.4 SCHEMATIC DIAGRAM**



### 3.5 MAIN COMPONENTS/ MODULES USED

#### Node MCU V3 ESP8266 WiFi Module

32 bit microcontroller with in-built Wifi and 4MB flash memory and CH340G USB interface for programming and file upload.

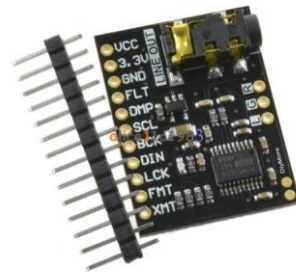
It is essentially a 3 volt device and the pins are not 5 volt tolerant. It can supply 5volt devices through the pin VU which is directly connected to the USB power pin.



#### PCM5102A DAC Decoder I2S 32bit Player Module

This device generates the audio signal. It supports the 16bit depth/ 16kHz sample rate WAV encoding used in this project in a simple manner. It can support other bit depths and sampling frequencies but some require an additional clock signal on pin SCL.

It delivers an up to 2 volt line output. There is an onboard jack but do NOT use this for low impedance headphones etc. or you may damage the device (as I have already done). There are no capacitors in the output path so it is more vulnerable to overloading or short circuits. Alternative I2S DACs are available. Check, however, that these support the bit depth, sample rate, your choice of output device (some support only loud speakers), generate their own system clock, and, if required, have prototype board (or even breadboard) compatible pin spacing.



Pinout:

VCC: 5 volts  
3.3V: derived from onboard regulator.  
GND: Common Ground  
FLT: Filter Latency [normal: LOW]  
DMP: De-emphasis [off: LOW]  
SCL: System Clock (not used)  
BCK: Bit clock  
DIN: Data Input  
LCK: Left/Right clock  
FMT : Format [I2S=LOW]  
XMT: /Mute [No Mute= HIGH(3.3v)]  
L: Left channel out  
G: Common Ground  
R: Right channel out

## RTC DS3231

This is a precision clock module. In this application, it is configured additionally to deliver a pulse on pin SQW at each seconds rollover for sub second accuracy.

These are 3 or 5 volts compatible.

Note: these are intended to be used with rechargeable cells (LIR2032) for backup. If you wish to use cheaper CR2032 non-rechargeable cells, you must disable the charging circuit say by removing the resistor directly above the SCL label by the 4-hole group on the board.



## 1602 16x2 LCD Character Display + IIC/I2C/TWI/SPI Serial interface Board Module

As I2C devices, these require only 2 wires in addition to the power wires.

It has a comprehensive built-in font set and the user can define a limited number of special characters. This feature has been used to define symbols such as for “WLAN connected”.

For this project, the 4 projecting pins on the backpack module have been modified to reduce the overall width of the combined module.



No backlight dimming is provided in this project.

The one used in this project is a 5 volt device, therefore a logic level shifter has been used to match it to the ESP8266. 3 volt 1602 devices are also available.

## IIC I2C Logic Level Converter Bi-Directional Module 5V to 3.3V

These are necessary for using 5 volt I2C devices with the ESP8266. Note: some similar designs (where the pins may be marked TX0, RX0 etc.) do not have a dedicated mosfet per pin pair (eg LV1/HV1) and are not suitable for I2C.



### **TDA2822 2X1.5W Dual Channel Audio Amplifier Module**

This is optional. This model is easy to use because all grounds are common. It is stereo, but only the left channel is used in this application.

It is switched on as required by the ESP8266 through a separate high side switch consisting of an N and a P channel mosfet. (see schematic diagram)



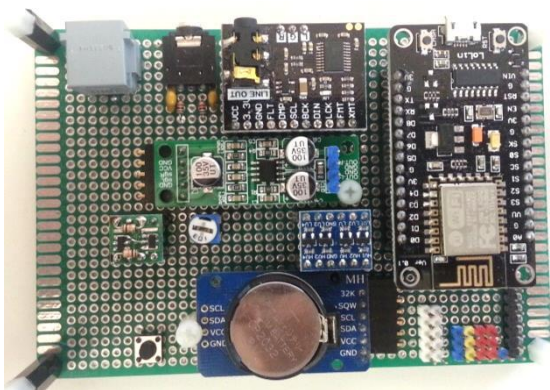
### **40mm Bass Multimedia Loudspeaker 3W 40hm Audio Speaker Radio ZB59009**

This is optional. The mounting holes make it easy to install.



## **4 CONSTRUCTION**

No printed circuit board design has been produced for this project and no specific layout is prescribed so this chapter is simply a collection of notes.



The pictures show a suggested module layout and for the interconnections, refer to the schematic diagram and markings on the modules.

A double sided 12cm X 8cm (0.1" hole spacing) prototype board has been used. For connections on the rear of the board, 32 AWG polyurethane insulated wire and uninsulated tinned 30AWG wire used.

The modules are not soldered directly to the prototype board, but are mounted in cut strips of 0.1" female header pins. This allows reuse of components and easy trouble shooting by swapping out suspect components.

A small number of discrete components are used. The 2 mosfets for the amplifier switch are SMD components and I have used a small break out board for these.

The RJ11 telephone socket does not have a pinout which is compatible with the prototype board so a sufficiently large slot was milled for the pin group to project through.

The loud speaker, when mounted as illustrated requires only two small (4 mm) holes for the sound port towards the outside for maximum volume, although 5 are shown.

## 5 USER MANUAL

### 5.1 INITIAL SET UP

This describes the basic set up of an unconfigured or 'factory condition' device.

- Create a directory for the software from the deployment pack
- Add a blank directory to it called data
- In the new directory data, copy a matched pair of .wav and .trk files.
- In the Arduino IDE, locate the .ino file from the deployment pack to open the project. (The Arduino IDE must be loaded with the required code libraries, the appropriate ESP8266 board type selected , SPIFFS file size 3MB and 160 MHz clock selected)
- Use the tool ESP8266 sketch data upload to transfer the contents of the data directory to the SPIFFS file sytem
- Compile and load the sketch to the ESP8266 then open the serial console at 115200 baud.
- The device will be in forced into AP mode, so connect your smartphone or WLAN capable PC to the SSID in the display (by default SpeakCL)

- Open a web browser and enter the IP Address seen on the display (by default 192.168.4.1)
- Follow the steps in “Web Configuration” below to enter the credentials of your WLAN (SSID and PSK) and timezone information.
- Power the device off, then on again and it should, after a few seconds, fetch the correct time from the configured NTP server.
- When the device is joined to your WLAN, you can now contact it using the IP address which can be displayed on the LCD screen by pressing the control button for more than 8 seconds.

## 5.2 WEB CONFIGURATION

Web configuration mode is entered by typing the device’s IP address into a web browser in the same WLAN.

If the device has not joined an existing WLAN, i.e. the device is in AP Mode, then it will be first necessary to connect the smartphone or WLAN capable PC to the network created by the device.

You can navigate between pages by selecting the page number from the top bar and make changes. Press ‘Update’ when you have completed your changes.

### Configuration page 1

This shows the software version, date and time stamps at the last browser refresh. The sync status is OK if the device clock has got a valid time stamp from one of its sources.

NTP Sync Age is OK if the last NTP Sync was less than X minutes ago, otherwise a ‘!’ is shown. The same with RTC status. Since NTP has a higher priority than the RTC, a ‘!’ is usually shown for the RTC if the NTP is ‘Ok’.

The device name in bold at the top, in this case ClockSP can be changes by the user. It is also the name of the SSID when the device is in AP mode.

ClockSP	
1	2 3 4 5 update
Status	
Version:	spck_v0.47
Datestamp:	20-07-2018 (Fri)
Timestamp:	00:29:51
Sync Status:	Ok
NTP sync Age:	Ok
RTC Status:	!

## Configuration page 2

The clock name can be changed. It cannot be blank.

An SSID and PSK (password). These must be present even if there is no local WLAN. Just use the word 'dummy'

Once a PSK has been registered in the system, there is no need to reenter it each time you make a change on the page.

If no WLAN is available, click the box. Since NTP will not then work, it will then be necessary to set the time manually on page 3.

The default time server and sync interval are shown here. If you wish to disable NTP sync, set the time interval to 0.

Errors appear like this on update, in this case the clock name cannot be blank.

## ClockSP

1 2 3 4 5 update

System Name	
Clock Name:	<input type="text" value="ClockSP"/>
WiFi Setup	
SSID:	<input type="text" value="Born-to-Run"/>
PSK:	<input type="password" value="•••••"/>
Wlan not available: <input type="checkbox"/>	
Wifi required for NTP sync disabled WiFi can be restored using AP mode	
Time Setup	
NTP Server:	<input type="text" value="0.ch.pool.ntp.org"/>
Sync Interval:	<input type="text" value="64"/> sec
set Sync Intervat to 0 to disable sync	

## ClockSP

1 2 3 4 5 update

invalid clockname.

System Name	
Clock Name:	<input type="text" value=""/>
WiFi Setup	
SSID:	<input type="text" value=""/>

### Configuration page 3

If you need to set the time manually, do it according to the example format here then click the checkbox and then update.

If NTP is left active (page 2) then any time you set manually will be overridden at the next NTP synchronization.

5 predefined timezones are configured. If yours is not in the list, select custom then enter the rules for your timezone as illustrated below.

On selection of Custom timezone, you see a panel similar to this.

In this example, standard time (STD) begins at 03:00 am on the last Sunday in October and the offset from UTC is +60 minutes.

Daylight Saving Time (DST) begins at 02:00 am on the last Sunday in March and the offset from UTC is +120 minutes.

## ClockSP

1 2 3 4 5 update

### Manual Time Setup

Time (local): 01:04:50

Date (local): 20-07-2018

Update Date/Time: ☐

disable NTP to prevent this setting being overwritten.

### Timezone

TZ Select: CET/CEST

CET/CEST

GMT/BST

usEST/EDT

usCST/CDT

UTC

Custom

### Timezone

TZ Select: Custom

### Custom TZ

	week	day	mon	hour	offset mins
STD:	Last	Sun	Oct	3	60
DST:	Last	Sun	Mar	2	120

## Configuration page 4

This gives data about the current audio file and matching track (index) file.

The volume level is the input to the dac. Adjust this first so that it is correct for a connected telephone or external audio device.

Changing this will also affect the input to the internal amplifier (if fitted).

Play time is the length of time the amplifier will be switched on for on pressing the control button. It will always terminate automatically at the end of a completed cycle.

## ClockSP

1 2 3 4 5 update

Voice App Data	
<b>Track File</b>	
Name:	/patSimmons_v0.01.c.trk
Description:	GB_PSimmons_12H
Speech Workflow:	GPO_FIX_12H
<b>Audio File</b>	
Name:	/patSimmons_v0.01.wav

Voice App Config	
Volume factor:	<input type="text" value="32"/>
dac input 0-255 default 32	
play time:	<input type="text" value="30"/>
Seconds. Terminates at end of cycle	

## Configuration page 5

This is intended for advanced level troubleshooting.

Selecting debug shows the data transmitted between the browser and the web server

## ClockSP

1 2 3 4 5 update

debug: ☒

```
{"updateResponse":"","cpa_versionNr":"spck_v0.47","cpa_serverDateLocal":20-07-2018 (Fri); 01:04:50 ; output;
```

```
updateResponse:;; output;
```

```
cpa_versionNr: spck_v0.47; spck_v0.47; output;
```

```
cpa_serverDateLocal: 20-07-2018 (Fri); 20-07-2018 (Fri); output;
```

```
cpa_serverTimeLocal: 01:04:50 ; 01:04:50 ; output;
```

```
cpa clockName: ClockSP: ClockSP: text;
```

## 5.3 LCD SCREEN

This is used for showing time, date and status information. Further messages may appear during error or configuration and these will be referred to in the appropriate chapter in this document.

Normal Mode showing time, date, day and symbols representing the state of the device.

```
20-07-2018 (Fri)
00:09:15      #T<
```

The symbols on the bottom right from left to right (magnified)

1. NTP Synchronisation OK
2. WLAN connected
3. Loudspeaker active



If any of the above conditions are not met, the symbol is replaced with a placeholder as shown here.



## 5.4 CONTROL BUTTON

The control button has a number of functions depending on the state of the system when it is pressed and the length of time it is pressed for. It is active on release.

### 5.4.1 TOGGLE AMPLIFIER ON/OFF

During normal operation, a short press will switch the amplifier on for a preset length of time, terminating automatically at the end of an announcement cycle. If the amplifier is already on, a short press will immediately switch it off.

### 5.4.2 DISPLAY CONFIGURATION INFORMATION ON THE LCD SCREEN

Holding the button for 8 seconds or more before release cause a set of configuration information to be shown on the screen, paging at 3 second intervals:

- Software Version
- SSID (wireless lan network ID)
- IP Address in network (if connected).
- NTP/RTC Status
- Audio file short display name.

### 5.4.3 FORCE DEVICE INTO AP MODE

If the button is held in during system start for at least 8 seconds (but less than 20 seconds!), the device is put into AP Mode, that is it creates its own wireless lan and the SSID and IP address are shown on the LCD screen. The device remains in the mode until it is disconnected from the power source.

This is normally used for the initial configuration of the device through a smart phone or WLAN capable PC.

### 5.4.4 FORCE FACTORY RESET

If the button is held in during system start for at least 30 seconds, the device will go into a factory reset mode and all user configuration is erased. Any WLAN credentials which were entered into the device, or any other nonstandard configuration has to be re-entered in AP mode. This may be used in advanced trouble shooting, or for certain data privacy reasons.

## 5.5 FILE FORMATS

A pair of files is required to generate an audio announcement and these are copied into the ESP8266 flash memory.

### 5.5.1 WAVE (.WAV) FILE

This contains all phrases and numbers from a voice actor concatenated together with a short intervening pause.

The file name is up to 25 characters including the mandatory lower case extension '.wav'. e.g. **patSimmons\_v0.01.wav**

It must be encoded with a 16 bit bit depth and 16kHz sample rate PCM. Audacity or some online tools can be used to convert wave file formats.

Note. If you edit an existing .wav file to alter individual parts of the spoken text, attempt not to alter the relative positions of the following spoken text otherwise you may have to make a considerable number of changes to the matching .trk file.

### 5.5.2 TRACK (.TRK) FILE

This is an index into the .wav file and contains a set of header lines and one line per phrase or per number indicating its start and end times in units of a hundredth of a second relative to the beginning of audio file. Here is an extract.

```
speaking clock track descriptor. See manual for format. No embedded blanks in tokens
GPO_FIX_12H                ; workflow name (Max 12 chars)
/patSimmons_v0.01.wav      ; wav audio file described by this file (max 25 chars)
GB_PSimmons_12H           ; track display name (max 16 chars)
00575 00589                ; phrase 0: beep
06179 06469                ; phrase 1: silence
00051 00277                ; phrase 2: at the third stroke it will be
00365 00398                ; phrase 3: and
00404 00479                ; phrase 4: seconds
00283 00355                ; phrase 5: o'clock
00486 00569                ; phrase 6: precisely
00575 00589                ; phrase 7: -
00575 00589                ; phrase 8: -
00575 00589                ; number 0 -
00657 00709                ; number 1
00716 00765                ; number 2
00772 00832                ; number 3
00838 00885                ; number 4
...
...
00004 00018                ; (66) spare
00004 00018                ; (67) spare
00004 00018                ; (68) spare
00004 00018                ; (69) spare
```

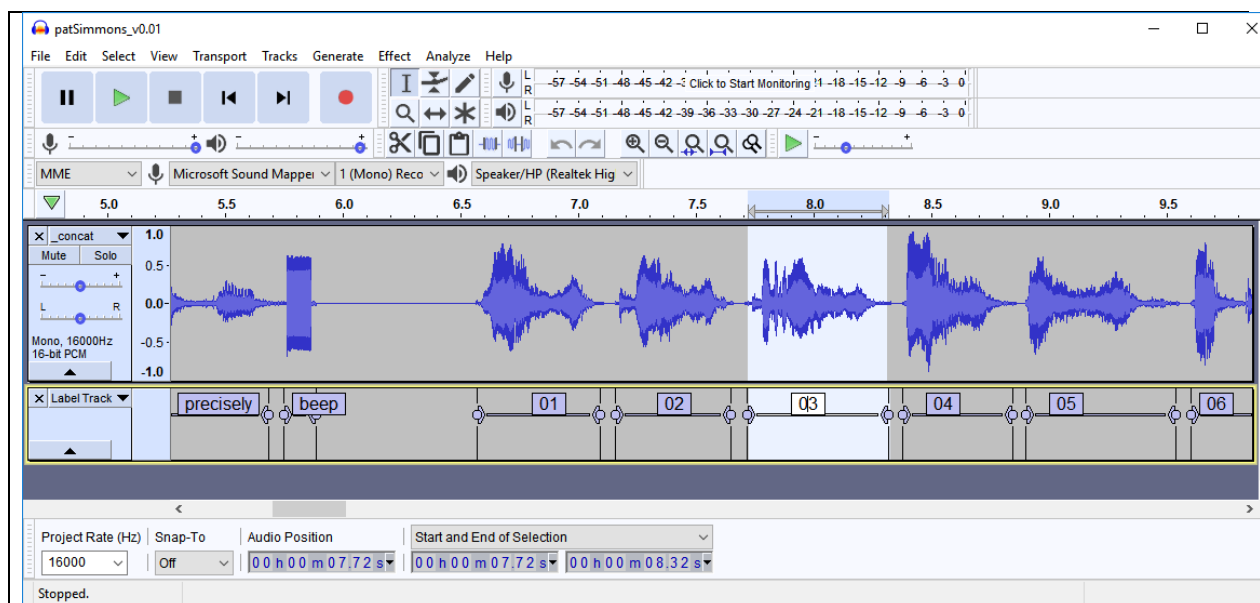
The header consists of 4 lines. Only the first token on the line is used.

- Line 1 is reserved for comments and must exist even if blank.
- Line 2 is the name of the workflow which controls the order and timings of the announcements. It must already be defined in the system.
- Line 3 is the audio file name preceded with a '/' character
- Line 4 is a user chosen short name for the audio set

The lines following the header consist of 2 tokens and a comment. That is the start and end of a number of a phrase. The order that phrases and numbers appear in is important to the specific workflow used.

For example, the number 3 in the .trk file above has the start time of 772 and the end time of 832.

Below is a section of the wav file as presented in Audacity. The audio section for the spoken number 3 is highlighted.



In the box entitled Start and End of Selection, the start and end of the section, that is for the spoken number 3, are 7.72 and 8.32 seconds respectively and the entry in the track file matches this.

In summary the track file is built up of 4 header lines, 9 phrases, 60 numbers and 10 spare entries.

## 5.6 CREATING VOICE RECORDINGS

Instructions on producing decent voice recordings are outside the scope of this document, but many tutorials can be found e.g. [Ref: 2].

In principle, Audacity can be used to record or process an existing recording, stripping out excess silence between the spoken words and altering characteristics such as sample rate and bit depth.

The order of activities is :

1. create a .wav file with all the required phrases and numbers concatenated together with a short say 0.1 second silent space in between. The order is that these appear in the file is not important.
2. select an existing workflow or create a new one (see below).
3. analyse the .wav file in Audacity and create a .trk file (probably based on a copy of an existing one ). The order of appearance of the phrases and numbers in the .trk file must match the workflow. Add the start and end time (in units of 1/100 of a second) of each phrase and number.

## 5.7 CREATING WORKFLOWS

The workflow defines the timings within the 10 second cycle specifying what phrase or number should be passed to the audio player at what time. Workflows are currently represented as C++ functions in the Speech Engine section of the program and adding or changing these requires a recompilation of the software.

If an existing workflow does not match your needs, you should create a new one and best start by making a copy of an existing one. If you simply edit an existing one, you risk rendering it invalid for previous recordings.

There are comments within the source code informing of the main points to observe. However, creating workflows is a topic for an advanced user and no further instructions appear in this document.

## 5.8 ERROR NUMBERS

The following can appear during the validation by the speech engine of the input files

- 1 Spiffs file system error
- 3 failed to find track file
- 4 failed to open track file
- 5 unsupported workflow
- 6 failed to find audio file
- 7 premature end of phrases
- 8 premature end of numbers
- 11 Directory must contain exactly one .trk file

## 6 POSSIBLE FUTURE DEVELOPMENT

Migrate to ESP32

Larger flash memory for multiple voice files

Load flash memory from SD Card instead of the USB load mechanism-

## 7 TROUBLE SHOOTING TIPS.

1. If your hardware choice, particularly the ESP8266 Module or the DAC , is different to that specified, suspect that first.
2. If you have made any changes to the software, always revert to the last known good version. Note: The design is particularly sensitive to any blocking code because the internal audio buffers have to be replenished every few milliseconds.
3. Look at the output on the serial console (115200 Baud) for indications of an error state. Errors which appear on the LCD screen usually have a verbose equivalent on the serial console.
4. Add additional error messages as required within the source code and recompile.
5. You can comment out selected function calls from the loop of the .ino file to disable selected modules in an attempt to isolate the problem.

## 8 SOFTWARE DISTRIBUTION

Location: <http://forum.arduino.cc/index.php?topic=559652.0>

The Audio file sets, because of their size, are at a different location referenced at the software distribution location.

Tested with the following, but others could work:

- Arduino IDE v 1.82
- Arduino ESP8266 Core Software 2.4.0
- Spiffs File Upload 0.2.0
- Lolin NodeMCU V3 ( 4MB flash)

## 9 APPENDIX

A document collection, some directly referred to above or used in the solution and some generally relevant further reading.

[Ref: 1] Wave file header <http://soundfile.sapp.org/doc/WaveFormat>

[Ref: 2] Creating voice files. <https://www.youtube.com/watch?v=diFvaF9yNd0> (one of many)

[Ref:3] ESP8266 BBX10 audio sound example <https://github.com/bbx10/SFX-I2S-web-trigger>

[Ref:4] The THG's TIM2015 Speaking Clock. A description focusing on building the device including some tips for producing quality sound recordings:  
[http://www.samhallas.co.uk/repository/tim\\_2015.htm](http://www.samhallas.co.uk/repository/tim_2015.htm)