**Documentation of extra feature**

As a special feature we implemented an alarm clock. Therefore we implemented a quartz for precise time counting and a speaker for the wake up noise. This setup allows for all kinds of time related applications, but due to the lack of time its currently used for a kitchen clock only.

**Software**

**Microcontroller**

*(The code is in the appendix)*

**Change code to fit quartz frequency**

The module “alarm.c” and its headerfile “alarm.h” have two tasks to perform, counting seconds and generate the alarm tone signal for the speaker.

Counting seconds is done with an external quartz (32.768kHz) acting as clock for the asynchronous timer/counter. With prescaler and compare value register the resulting frequency will be 1Hz and the interrupt service routine can be used to count seconds.

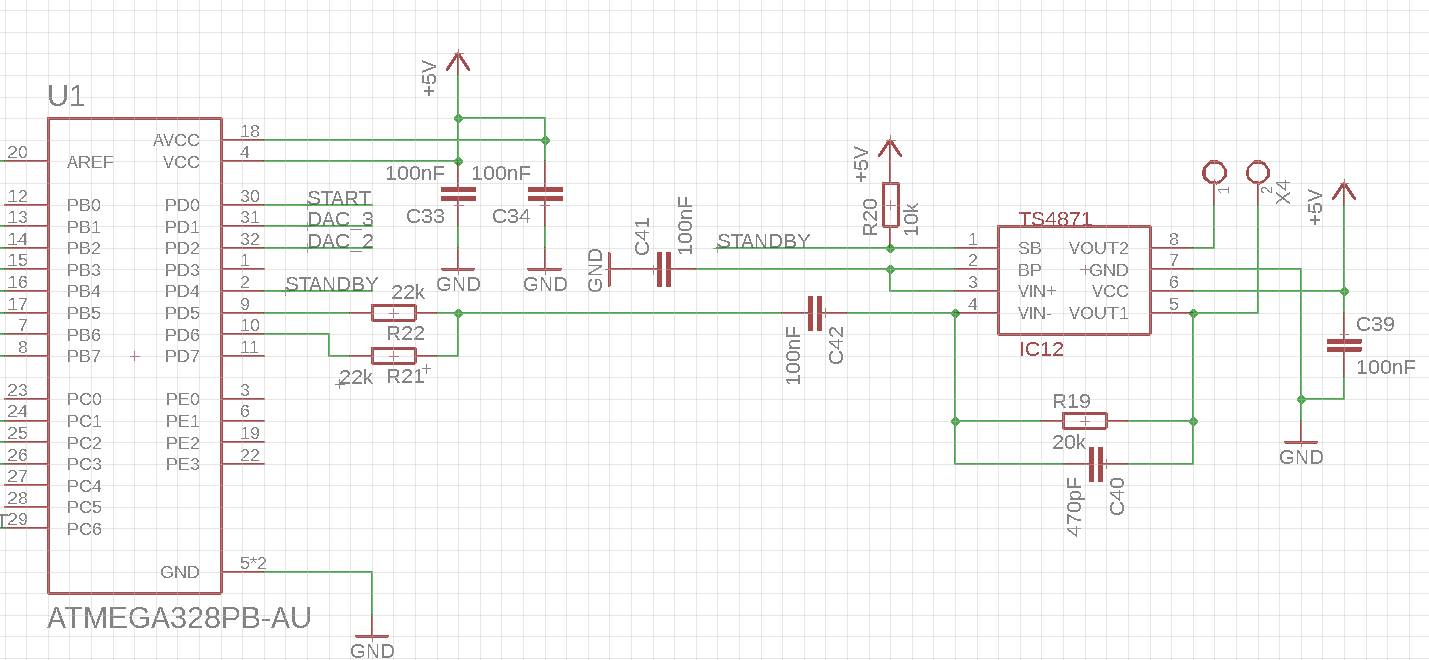
Waveform generation is realised with an 8-bit timer/counter in phase correct mode. The counter counts up and down. The two compare outputs are configured to be low when their compare value is lower than the counter value. This setup is always running. On and off for the alarm sound is done via the standby input of the audio amplifier.

**Android**

The alarm part of the Android app is yet quite simple. The affiliated activity contains a textView to display the set alarm time and a button to open a time picker dialog and set a new alarm time.

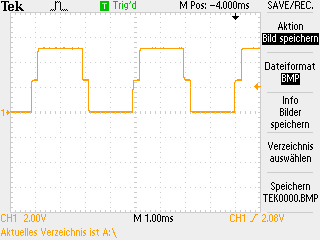
Hardware of the alarm clock

The function of the hardware part of the alarm clock is to combine amplify the signals from the microcontroller to drive the speaker of the alarm clock.



Schematic of the speaker circuit

The two PWM signals generated by the microcontroller on pins PD5 and PD6 are combined with the resistors R22 and R21. After merging the two signals the waveform looks like shown below.



Capacitor C42 blocks DC signals from reaching the amplifier circuit.

IC12 is a 1-Watt audio amplifier to drive the speaker of the alarm clock. R20 is a pullup Resistor. If the STANDBY signal is pulled low the amplifier is in operation otherwise the amplifier is in standby to save energy. Capacitor C40 and Resistor R19 together form a high pass filter to block frequencies over 17 kHz.   
The loud speaker can be connected to the terminal block X4.

**public class** AlarmActivity **extends** AppCompatActivity **implements** TimePickerDialog.OnTimeSetListener, AlarmRequestTimerListener {  
  
 *// constants* **final private int ALARM\_UPDATE\_TIME\_MS** = 10000; *// = 10s  
  
 // alarm update timer* AlarmRequestTimer **timer**;  
  
 *// GUI instances* TextView **timeText**;  
 Button **timeButton**;  
  
 @Override  
 **public void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_alarm***);  
  
 *// GUI initialization* **timeText** = (TextView) findViewById(R.id.***timeText***);  
 **timeButton** = (Button) findViewById(R.id.***timeButton***);  
  
 **timeButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *// create and open a new time picker dialog* android.support.v4.app.DialogFragment timePicker =  
 **new** TimePickerFragment();  
 timePicker.show(getSupportFragmentManager(), **"timepicker"**);  
 }  
 });  
 }  
  
 @Override  
 **public void** onStart() {  
 **super**.onStart();  
 *// create a new timer to request the current alarm time in the set intervall* **timer** = **new** AlarmRequestTimer(**this**, **ALARM\_UPDATE\_TIME\_MS**);  
 }  
  
 @Override  
 **public void** onStop() {  
 **super**.onStop();  
 *// kill any existing timer on activity stop to avoid left behind threads* **try** {  
 **timer**.kill();  
 } **catch** (Exception e) {  
 *// necessary to avoid errors if the thread doesn’t exist* }  
 }  
  
 */\*\*  
 \* overrides the onTimeSet function of the OnTimeSetListener  
 \** ***@param view*** *\** ***@param hourOfDay*** *\** ***@param minute*** *\*/* @Override  
 **public void** onTimeSet(TimePicker view, **int** hourOfDay, **int** minute) {  
 **int** seconds = (hourOfDay \* 3600) + (minute \* 60);  
 **timeText**.setText(timeToString(hourOfDay, minute));  
 sendAlarmTime(seconds);  
 }

*/\*\*  
 \* sends the moodlight a request to send the current alarm time back  
 \* calls sendAlarmTimeRequest()  
 \*/* **public void** timeElapsed() {  
 sendAlarmTimeRequest();  
 }  
  
 */\*\*  
 \* void sendAlarmTimeRequest(void)  
 \* sends a request to moodlight to get the current value of the alarm timer  
 \*/* **private void** sendAlarmTimeRequest() {  
 **byte**[] buffer = **new byte**[7];  
 buffer[0] = MainActivity.***BT\_ALARM***;  
 buffer[1] = MainActivity.***BT\_REQUEST***;  
 buffer[2] = 0x00;  
 buffer[3] = 0x00;  
 buffer[4] = 0x00;  
 buffer[5] = 0x00;  
 buffer[6] = MainActivity.***BT\_DELIMITER***;  
 MainActivity.*bt*.send(buffer, **false**);  
 }  
  
 */\*\*  
 \* void sendAlarmTime(int)  
 \* sends a new value to the moodlight  
 \** ***@param value*** *\*/* **private void** sendAlarmTime(**int** value){  
 **byte**[] buffer = **new byte**[7];  
 buffer[0] = MainActivity.***BT\_ALARM***;  
 buffer[1] = MainActivity.***BT\_SEND***;  
 buffer[2] = (**byte**) ((value >> 24) & 0xFF);  
 buffer[3] = (**byte**) ((value >> 16) & 0xFF);  
 buffer[4] = (**byte**) ((value >> 8) & 0xFF);  
 buffer[5] = (**byte**) ((value >> 0) & 0xFF);  
 buffer[6] = MainActivity.***BT\_DELIMITER***;  
 MainActivity.*bt*.send(buffer, **false**);  
 }  
  
 */\*\*  
 \* String timeToString(int, int)  
 \* converts to values (hours and minutes) into a String  
 \* of the format "hh : mm";  
 \** ***@param hours*** *\** ***@param minutes*** *\** ***@return*** *\*/* **private** String timeToString(**int** hours, **int** minutes){  
 String text = **""**;  
 **if**(hours < 10){  
 text += **"0"**;  
 }  
 text += hours;  
 text += **" : "**;  
 **if** (minutes < 10){  
 text += **"0"**;  
 }  
 text += minutes;  
 **return** text;  
 }  
}