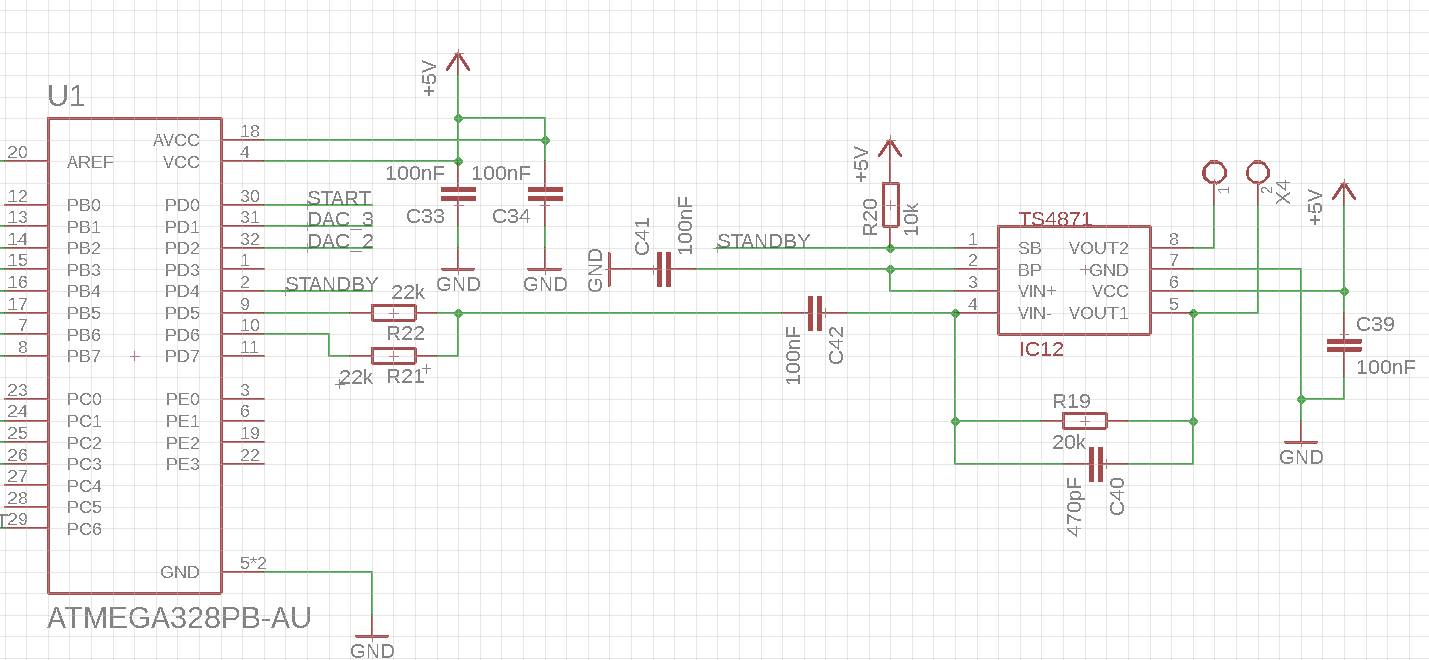
**Documentation of extra feature**

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.

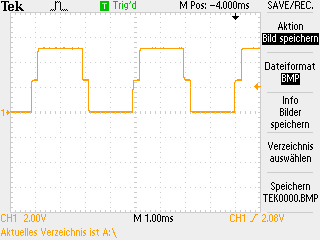
**Hardware of the alarm clock**

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.

**PCB Layout**

Don’t place the circuit near a high power circuit to minimize external influences.  
Place the filter capacitors of the ICs close to the IC.

**Software**

**Microcontroller**

*(The code is in the appendix)*

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.

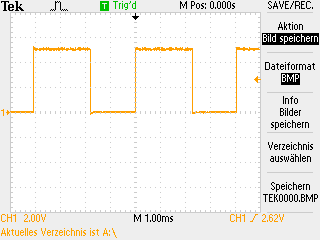
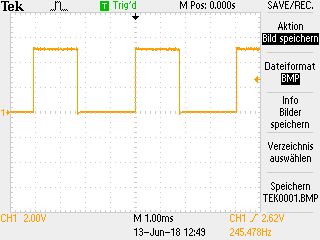
**Android**

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.

When starting the AlarmActivity, the app sends a request to the moodlight for it to send back the current timer value. Unfortunately, this doesn’t work reliably yet. On clicking the "edit"-button the app creates and shows a new time picker dialog. This dialog allows to choose a time in hours and minutes (max. 23:59). After the set time the alarm will start and after another 20 seconds it will stop again.

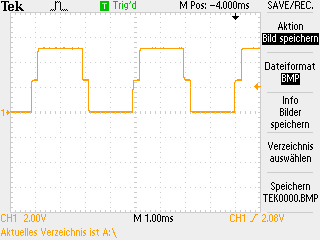
**Test procedure**

1. Plug in power supply
2. Check if 5V power is present on the amplifier (IC12).
3. Check if STANDBY signal to the amplifier is on HIGH (5 Volts) level. This should only be the case when the alarm sound is not played.
4. PD5 and PD6 (Pin 9 and pin 10 on U1) should each have a 245 Hz PWM. Each signal has a different duty cycle (Duty cycle depends on volume settings set in the software).



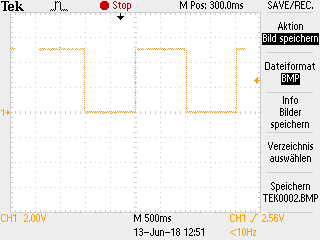
PWM on PIN 10 (PD6) PWM on PIN 10 (PD5)

1. Measure on C42 if the two PWM signals are combined correctly.



Both signals combined should look similar to the waveform above.

1. Set an alarm time in the Android app. After the set alarm time the alarm sound should start. When measuring the STANDBY signal it should be a rectangular waveform with 0.5 Hz. The alarm sound turns on and off every second.



0.5 Hz Signal of the STANDBY signal

**Appendix**

**Java Code**

**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***);  
  
 */\* set a new OnDataReceivedListener so it can use the local functions \*/* MainActivity.*bt*.setOnDataReceivedListener(**new** BluetoothSPP.OnDataReceivedListener() {  
 @Override  
 **public void** onDataReceived(**byte**[] data, String message) {  
 **if** ((data[0] == MainActivity.***BT\_ALARM***) && (data.**length** >=  
 MainActivity.***BT\_NORMAL\_MESSAGE\_LEN***)){  
 **int** alarmTime = 0;  
 alarmTime += (((**int**) data[2]) << 24);  
 alarmTime += (((**int**) data[3]) << 16);  
 alarmTime += (((**int**) data[4]) << 8);  
 alarmTime += (((**int**) data[5]) << 0);  
 setTimeText(alarmTime);  
 }  
 }  
 });  
  
 */\* 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) {  
 android.support.v4.app.DialogFragment timePicker = **new** TimePickerFragment();  
 timePicker.show(getSupportFragmentManager(), **"timepicker"**);  
 }  
 });  
 }  
  
 @Override  
 **public void** onStart() {  
 **super**.onStart();  
 *// create and start a new timer to update the time value regularly* **timer** = **new** AlarmRequestTimer(**this**, **ALARM\_UPDATE\_TIME\_MS**);  
 **timer**.start();  
 }  
  
 @Override  
 **public void** onStop() {  
 **super**.onStop();  
 *// destroy 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* }  
 }  
  
 */\*\*  
 \* Listener for the time picker dialog  
 \** ***@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();  
 }  
  
 */\*\*  
 \* converts seconds into hours and minutes and sets the text  
 \* of timeText  
 \* calls timeToString  
 \** ***@param seconds*** *\*/* **public void** setTimeText(**int** seconds){  
 **int** hours = seconds / 3600;  
 **int** minutes = (seconds % 3600) / 60;  
 **timeText**.setText(timeToString(hours, minutes));  
 }  
  
 */\*\*  
 \* sends a request to the moodlight to send back  
 \* the current timer value  
 \*/* **public void** sendAlarmTimeRequest() {  
 **byte**[] buffer = **new byte**[7];  
 buffer[0] = MainActivity.***BT\_ALARM***;  
 buffer[1] = MainActivity.***BT\_REQUEST***;  
 buffer[2] = (**byte**) (0);  
 buffer[3] = (**byte**) (0);  
 buffer[4] = (**byte**) (0);  
 buffer[5] = (**byte**) (0);  
 buffer[6] = MainActivity.***BT\_DELIMITER***;  
 MainActivity.*bt*.send(buffer, **false**);  
 }  
  
 */\*\*  
 \* sends the value [seconds] 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**);  
 }  
  
 */\*\*  
 \* converts the int 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;  
 }  
}

**C Code on the following pages**