

Multitasking Autotuning PID Controller in Heat Transfer Application

or how to build a 30 BGN universal vectorized regulator

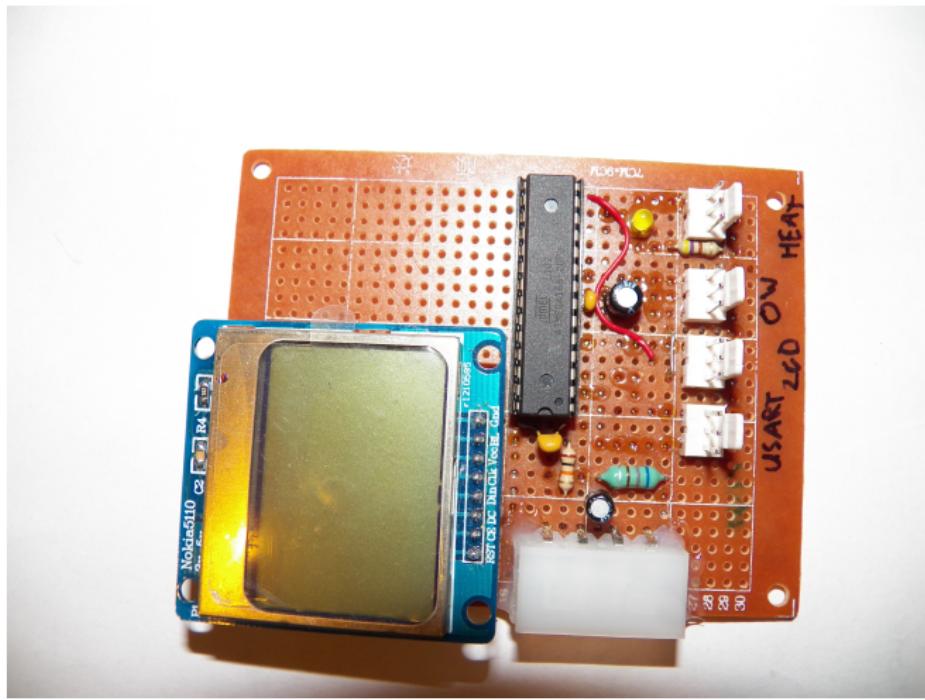
Miroslav Vitkov

Supervised by: Assoc. Prof. Alexander Ichtev

ELDE, Technical University-Sofia

January 26, 2016

The device



```
// Decide if this particular half-wave should be turned on. Further discussion:  
// http://programmers.stackexchange.com/questions/304546/algorithm-to-express-an-integer-as-a-sum-of-some-binary-numbers  
// The function simulates an uint16_t overflow.  
bool should_turn_on()  
{  
    static uint32_t acu = 0;  
    acu += g.setpoint;  
    if(acu > PROC_VAL_MAX)  
    {  
        acu &= PROC_VAL_MAX;  
        return true;  
    }  
    else  
    {  
        return false;  
    }  
}
```

Peak detector algorithm

```
pid_state_t pid_wait_to_settle(pid_inout_t proc_val, pid_inout_t critical, pid_inout_t threshold, clock_seconds_t now)
{
    extremum_t *min = &g_pid_tune.min;
    extremum_t *max = &g_pid_tune.max;
    extremum_t **curr = &g_pid_tune.curr;
    extremum_t **prev_ = &g_pid_tune.prev;

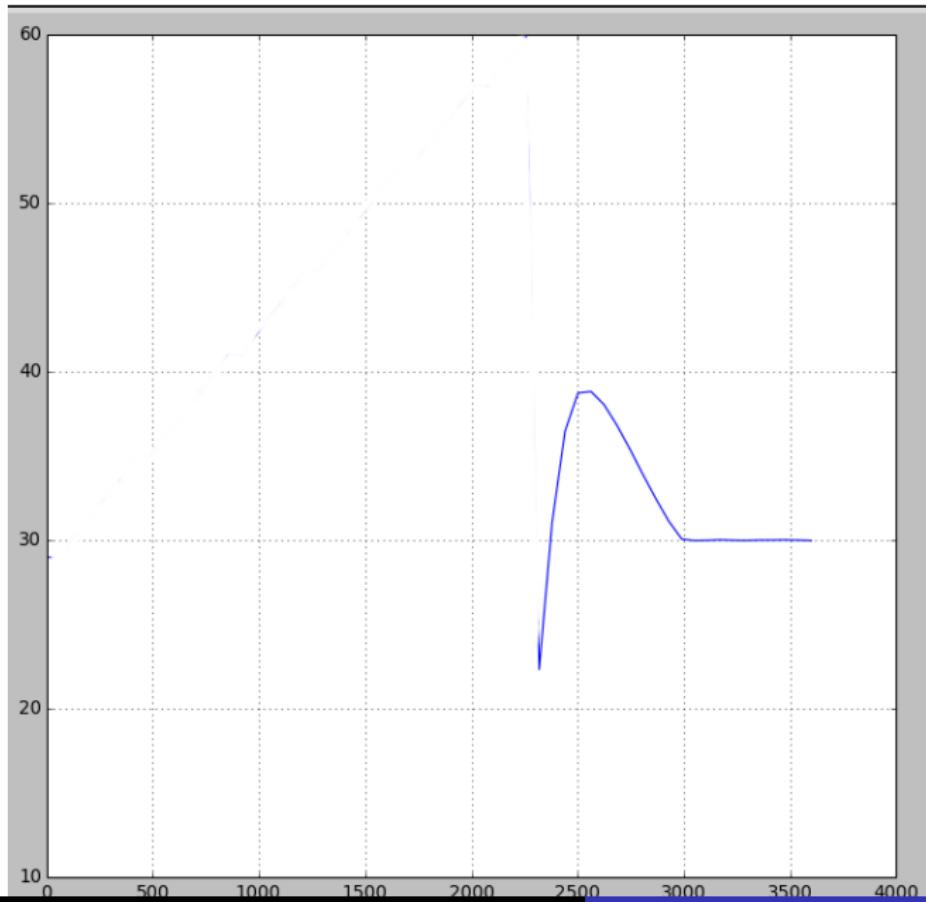
    static pid_inout_t prev, prevprev;

    if(*curr == min)
    {
        // Look for a local maximum.
        if(prevprev <= prev && prev > proc_val)
        {
            (*max).val = proc_val;
            (*max).when = now;
            *curr = max;
            *prev_ = min;
        }
    }
    else
    {
        if(prevprev >= prev && prev < proc_val)
        {
            (*min).val = proc_val;
            (*min).when = now;
            *curr = min;
            *prev_ = max;
        }
    }

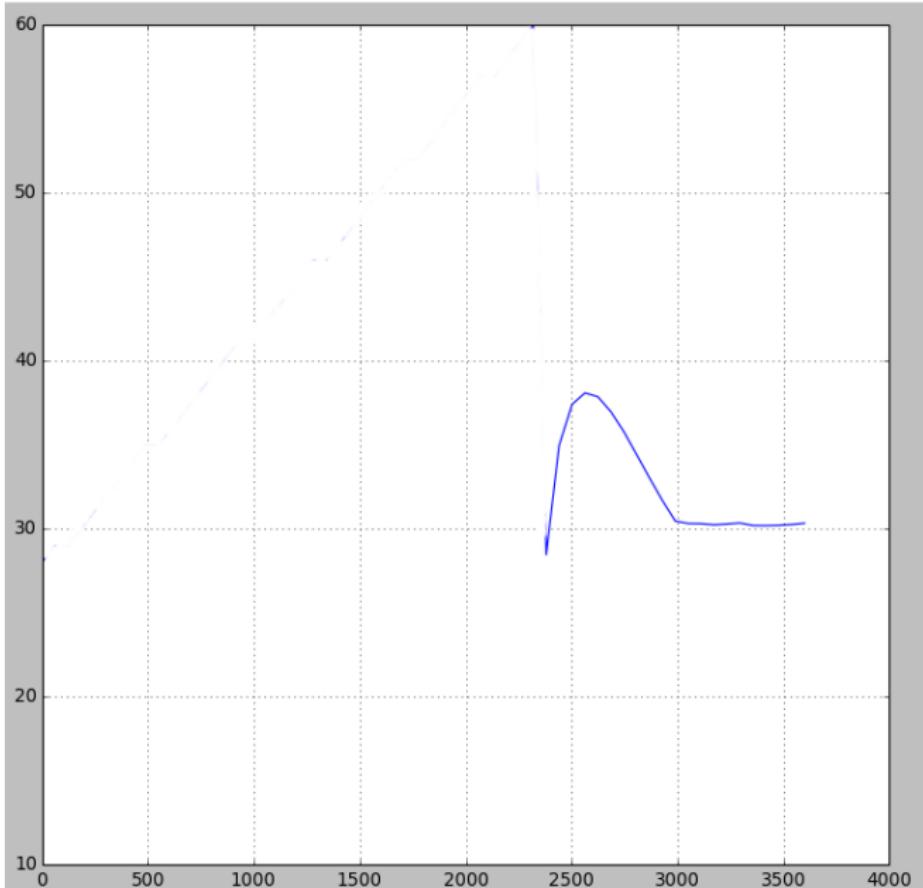
    prevprev = prev;
    prev = proc_val;

    if(proc_val >= critical) return UNSTABLE;
    else if((*min).val + threshold < (*max).val)
return SETTLED;
    else return OSCILLATING;
}
```

Ziegler-Nichols sustained oscillation mode



Astrom-Hagglund sustained oscillation mode



Sampling frequency reduced

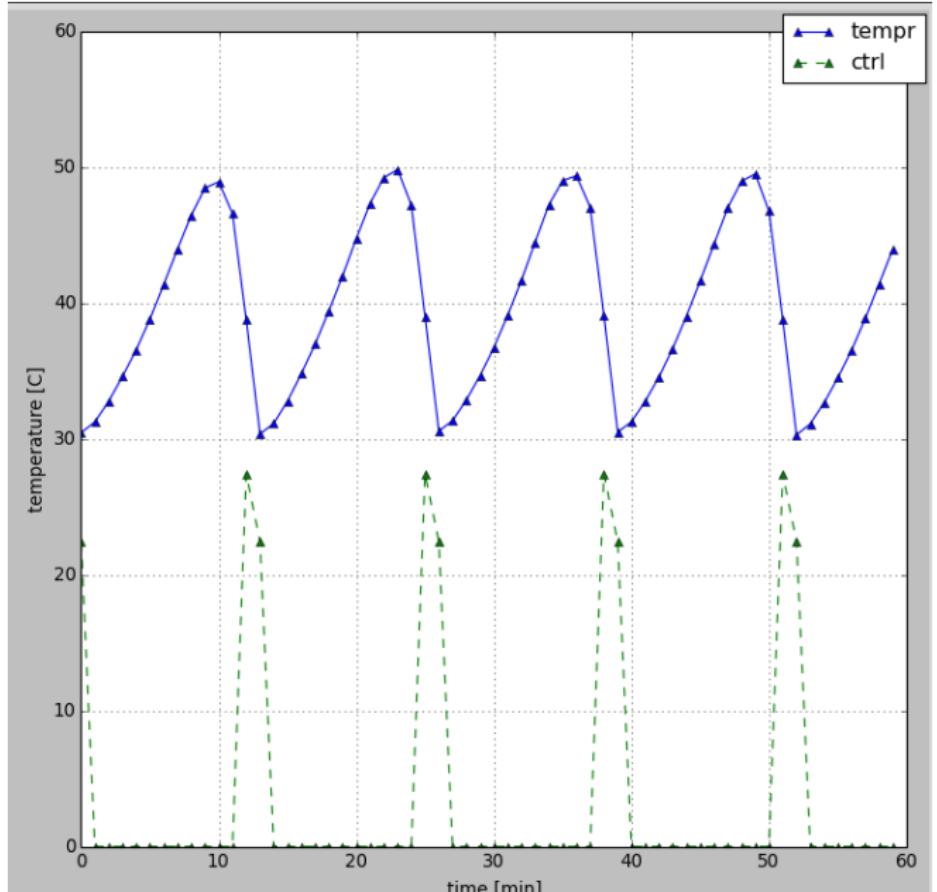


Table of Contents