```
/** EN-SOC3001
*  koden til prosjektet i ensoc3001
*/
#include "mbed.h"
#include "TextLCD.h"

//###
//# Config values:
//###
#define TMP_READ_INTERVAL 2.5//180.0
#define LCD_UPDATE_INTERVAL 1.0 //60.0

//###
//# input, output, interrupt and timer initalization.
//###
Ticker tempMaaling;
Ticker lcdUpdate;
Timeout killTimeOut;
Timeout lukeTimeOut;
Timeout debounce;
Serial pc(USBTX, USBRX);
InterruptIn userButton(PC_13);
InterruptIn resetBtn(D7);
TextLCD lcd(D11,D10,D9,D5,D4,D3,D2);
DigitalOut trapDoor(A0);
PwmOut killGrid(A1);
DigitalOut potPwr(A3);
AnalogIn potMeter(A4);
DigitalOut potGnd(A5);

//###
//# variables:
//###
double killDuration=2.0;
float killFreq=200.0;
int killCount=0;
char melding[80]; // Mesage to be displayed on the  LCD-Module
double temp = potMeter.read();
// character pointers for the parsed values from serial.
char* kommando;
char* verdi1;
char* verdi2;

//###
//# Prototpes
//###
void kill();
void killOff();
void luke();
void funk();
void lcdUpdater();
void reset();
void tempraturHent();
void pcTxSrialData();

int main()
{
    //###
    //# Print a useless boot screen.
    //###
    lcd.gotoxy(1,1);
    lcd.printf("Booting");
    lcd.gotoxy(1,2);
    lcd.printf("RatKill2000");
    for(int i=0;i<4;i++)
    {
```

```
        wait_ms(300);
        lcd.printf(".");
    }

    //###
    //# Initalizeing interrupts, and inputs.
    //###
    // Listen for serial data from the PC
    pc.attach(&pcTxSrialData);
    // Listen for button interrupt.
    userButton.fall(&kill);//&funk);
    resetBtn.fall(&reset);
    // ticker
    tempMaaling.attach(&tempraturHent,TMP_READ_INTERVAL);
    lcdUpdate.attach(&lcdUpdater,LCD_UPDATE_INTERVAL);
    // PWM
    killGrid.period( 1/killFreq); // 20ms Periode tid
    killGrid=0.00; //Set the ouput duty-cycle to 0%
    // set power and ground for the temp sensor/resistor.
    potPwr=1;
    potGnd=0;

    //###
    //# Clear the boot screen and add descriptors.
    //###
    lcd.lcdComand(0x01);// clear lcd.
    wait_ms(50);
    lcd.gotoxy(1,1);
    lcd.printf("Temp:");
    lcd.gotoxy(1,2);
    lcd.printf("Killcount: ");

    //###
    //# Main loop:
    //# (all funcions are interrupts, so the main is just to set in seep mode.)
    //###
    while(1)
    {
        wait_ms(300);
    }
}//end main



// Debug func, spits out the commands from serial.
void funk()
{
    printf(melding);
    printf("\n\r");
    printf(verdi1);
    printf("\n\r");
    printf(verdi2);
    printf("\n\r");
}

// reenable the detect interrupt after debounce timeout.
void enableDetect()
{
    userButton.enable_irq();
}

// activate the killgrid, it kills rats.
void kill()
{
    userButton.disable_irq(); // disable for debounce.
    debounce.attach(&enableDetect,0.005); // wait 0.005s for debounce.
```

```
    //activate kill grid.
    killGrid = 0.5; //set killgid to 50% dutycycle.
    killTimeOut.attach(&killOff,killDuration);
    //open trap-door
    killCount++;
    lcdUpdater();
}

// deactivate the killgrid.
void killOff()
{
    killGrid = 0.0;
    trapDoor=1;
    lukeTimeOut.attach(&luke,0.5);
}

//close the trap-door.
void luke()
{
    trapDoor=0;
}

// Update Values on the LCD.
void lcdUpdater()
{
    // this "place" variable and the if-tests make sure the temp
    // stays in the same spot on screen (LCD). and clears old characters.
    int place=12;
    if(temp>10||temp<0)
    place--;
    if(temp<-9)
    place--;
    lcd.gotoxy(place-2,1);
    lcd.printf("  ");
    lcd.gotoxy(place,1);
    lcd.printf("%.2fC",temp);
    lcd.gotoxy(12,2);
    lcd.printf("%d",killCount);
}

//reset the killCount, and update the display
void reset()
{
    killCount=0;
    lcd.gotoxy(12,2);
    lcd.printf("   ");
    printf("killCount,%d\r\n",killCount);
}

// Update the temp variable with the current temp.
void tempraturHent()
{
    temp=100.0*((double)potMeter.read()) - 50.0;
    //cast to double to avoid error.
}

// The pc has sendt a char to us, parse it.
void pcTxSrialData()
{
    scanf("%[^\r\n]s",melding);  // Read a whole line
    getchar(); // clear buffer.
    kommando = strtok(melding,","); // parse command.
    verdi1 = strtok(NULL,","); // parse value.

    // return temp.
    if(!strcmp(kommando,"getTemp"))
```

```
    {
        printf("temp,%f\n\r",temp);
    }

    // set kill settings.
    else if(!strcmp(kommando,"kill"))
    {
        killFreq=atof(verdi1);
        killGrid.period((float)(1.0f/killFreq)); // set frequency
        verdi2 = strtok(NULL,","); // parse value.
        killDuration = (atof(verdi2)); //set time in Seconds
        printf("killRead,%f,%f\r\n",killFreq,killDuration); // read frequency
    }

    // return kill settings.
    else if(!strcmp(kommando,"killRead"))
    {
        printf("killRead,%f,%f\r\n",killFreq,killDuration); // read frequency
    }

    //return amount killed.
    else if(!strcmp(kommando,"killCount"))
    {
        printf("killCount,%d\r\n",killCount);
    }

    //just a test command, so i don't actually have to leave my chair.
    else if(!strcmp(kommando,"drep"))
    {
        kill();
    }

    // reset the killcount. it's for emptying the trap.
    else if(!strcmp(kommando,"reset"))
    {
        killCount=0;
        lcd.gotoxy(12,2);
        lcd.printf("   ");
        printf("killCount,%d\r\n",killCount);
    }

    // Print a nondescriptive error message. (to maximize headache).
    else
    printf("nope\n\r");
}
```