Dílenská praxe

A4	3. Mikrovlnná trouba		
Petřík Vít		1/15	Známka:
23.10. 2019	Datum odevzdání:	27.11. 2019	Odevzdáno:

Zadání:

Zpracujte ovládací program v programovacím jazyce C ovládající model mikrovlnné trouby
tak, aby obsahoval nejméně tyto funkce:

1) simulujte provoz velmi jednoduché mikrovlnné trouby
2) na vestavěném displeji modelu zobrazuje jak dobu ohřevu, tak i teplotu „pokrmu“
3) na vestavěné klávesnici modelu umožní nastavit jak dobu ohřevu, tak i požadovanou teplotu.
4) na vestavěné klávesnici modelu umožní nastavit pracovní otáčky talíře
5) při „ohřevu“ průběžně zobrazuje metodou „countdown“ zbývající dobu ohřevu
6) sleduje a zobrazuje provozní a chybové stavy přípravku na monitoru PC

Postup:

- **Stav 0**
  - Výběr času ohřevu
- **Stav 1**
  - Výběr úrovně ohřevu
- **Stav 2**
  - Výběr úrovně výkonu motoru
- **Stav 3**
  - Odpočet do konce ohřevu
- **Stav 4**
  - Vypnutí bzučáku a návrat na stav 0

Propojení PC a Mikrovlnné trouby:

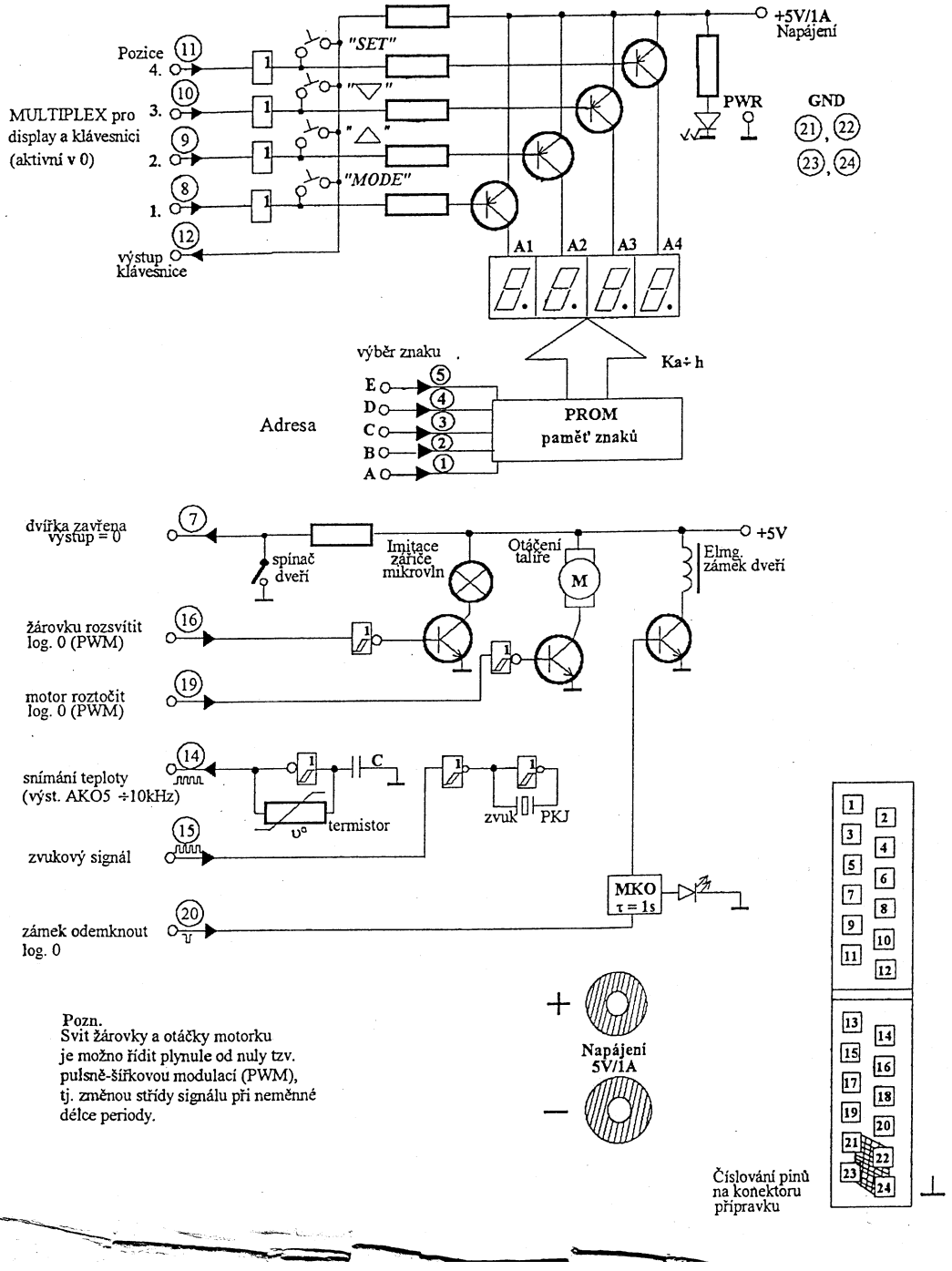
0x300 (IN) P1		0x301 (IN) P2		0x301 (OUT) P4	
Bit PC	**Pin MT**	**Bit PC**	**Pin MT**	**Bit PC**	**Pin MT**
0	A	0	8 (digit1)	0	-
1	B	1	9 (digit2)	1	-
2	C	2	10 (digit3)	2	-
3	D	3	11 (digit4)	3	-
4	E	4	16 (bulb)	4	-
5	-	5	19 (motor)	5	14 (temperature)
6	-	6	15 (tone)	6	7 (door)
7	-	7	20 (lock)	7	12 (key)

Schéma zapojení:

LABORATOŘE MIKROPROCESOROVÉ TECHNIKY

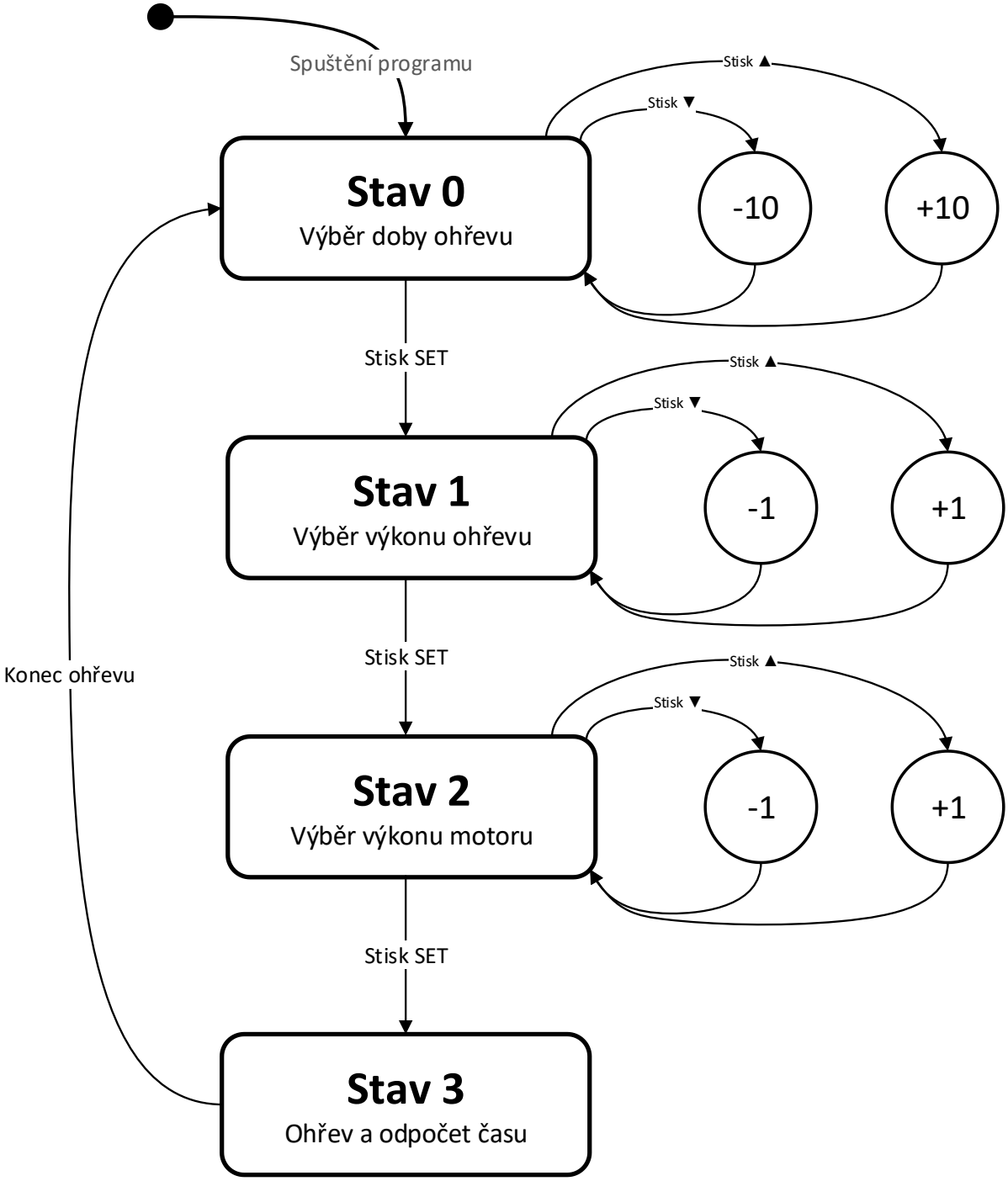
název úlohy: **Simulátor mikrovlnné trouby**

Schéma zapojení: *(čísla v kroužku odpovídají číslování pinů konektoru)*



Pozn.
Svit žárovky a otáčky motorku je možno řídit plynule od nuly tzv. pulsně-šířkovou modulací (PWM), tj. změnou střídy signálu při neměnné délce periody.

Vývojový diagram:



Výpis programu:

Příloha 1 – C++

Závěr:

Program funguje tak jak má a splňuje všechny body zadání. Při programování jsem využil znalostí OOP a stavových automatů, tak abych mohl oddělit řízení mikrovlnky od samotné logiky programu. Takový přístup mi vyhovuje, mám rád abstrakci a jsem si vědom, že při tvorbě složitějších programů/aplikací je OOP téměř nutností.

Přílohy:

- Příloha 1 – 9 stran

Příloha 1

```
/* ----------------------------------------------------------------
Zapojeni
- PortA (P1) (0x300) - OUT - selection displaying char (0x300)
    bit7    bit6    bit5    bit4    bit3    bit2    bit1    bit0
        -       -       -       E       D       C       B       A
- PortB (P2) (0x301) - OUT
    bit7    bit6    bit5    bit4    bit3    bit2    bit1    bit0
    digit1  digit2  digit3  digit4  bulb    motor   tone    lock
- PortC (P3) (0x300) - IN - sensors
    bit7    bit6    bit5        bit4    bit3    bit2    bit1    bit0
    key     door    temperature -       -       -       -       -
----------------------------------------------------------------*/

#include <time.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <string.h>

#define P1 0x300
#define P2 0x301
#define P3 0x300

#define LOCK_BIT 0
#define BUZZER_BIT 1
#define TURN_TABLE_BIT 2
#define BULB_BIT 3

#define TEMP_BIT 5
#define DOOR_BIT 6
#define KEY_BIT 7

//counter generovani PWM
#define COUNTER0_RESOLUTION 100
#define COUNTER0_PRESCALER 0

//counter generovani zvukoveho signalu
#define COUNTER1_RESOLUTION 2
#define COUNTER1_PRESCALER 100

//vrati pocet milisekund od startu programu
unsigned long millis()
{
    return (clock() * 1000) / CLOCKS_PER_SEC;
}

//funkce pro převod charu na kombinaci pro PROM
char charToByte(char c)
{
    switch (c)
    {
    case '0':
        return 0x00;
    case '1':
        return 0x01;
    case '2':
        return 0x02;
    case '3':
        return 0x03;
    case '4':
```

```
        return 0x04;
    case '5':
        return 0x05;
    case '6':
        return 0x06;
    case '7':
        return 0x07;
    case '8':
        return 0x08;
    case '9':
        return 0x09;
    case 'A':
        return 0x0A;
    case 'B':
        return 0x0B;
    case 'C':
        return 0x0C;
    case 'D':
        return 0x0D;
    case 'E':
        return 0x0E;
    case 'F':
        return 0x0F;
    case 'G':
        return 0x10;
    case 'H':
        return 0x11;
    case 'J':
        return 0x12;
    case 'L':
        return 0x13;
    case 'M':
        return 0x14;
    case 'N':
        return 0x15;
    case 'P':
        return 0x16;
    case 'R':
        return 0x17;
    case 'T':
        return 0x18;
    case 'U':
        return 0x19;
    case 'Z':
        return 0x1A;
    case '°':
        return 0x1B;
    case '.':
        return 0x1C;
    case '_':
        return 0x1D;
    case '-':
        return 0x1E;
    case '"':
        return 0x1F;
    default:
        return 0x1D;
    }
}

//třídá čítače pro generování PWM a audio signálu
class counter
```

```
{
private:
    int count, resolution, prescaler, prescalerCount;

public:
    //nastaveni citace
    void begin(int a, int b)
    {
        resolution = a;
        prescaler = b;
        count = 0;
        prescalerCount = 0;
    }

    //cyklus citace
    void run()
    {
        if (prescalerCount >= prescaler)
        {
            count++;
            prescalerCount = 0;
        }
        if (count + 1 > resolution)
        {
            count = 0;
        }
        prescalerCount++;
    }

    //vrati hodnotu citace
    int getCount()
    {
        return count;
    }
};

//třídá mikrovlnky která ovládá HW
class MikroVlnka
{
private:
    char doorClosed;
    int turnTablePWM;
    int bulbPWM;
    char buzzer;
    counter counter0, counter1;
    unsigned long lockMillis;
    unsigned long tempMillis;
    int tempFreq;
    int tempCounter;
    char index;
    char *string;
    char TEMP;
    char key;
    unsigned long keyMillis[4];
    unsigned long keyInterval[4];
    char keyValues[4];
    char keyLastState[4];
    char lastInput;

    //vyresi, jestli mi byt signal 0 nebo 1
    char solvePWM(int count, int PWM)
    {
```

```
        return ((count < PWM) ? 1 : 0);
    }

public:
    //nastavi porty na vychozi hodnotu (mikrovlnka ma aktivni stav na log. 0) a inizializuj
e citace
    void begin()
    {
        outportb(P2, 0x7F); //nastaví výstupní port na čtení prvního tlačítka
        tempMillis = millis();
        tempFreq = 0;
        tempCounter = 0;
        index = 0;
        setTurnTablePWM(0);
        setBulbPWM(0);
        setBuzzer(0);
        //nastavení proměnných pro kontrolu stisknutých kláves
        for (char i = 0; i < 4; i++)
        {
            keyMillis[i] = 0;
            keyInterval[i] = 500;
            keyLastState[i] = 0;
        }
        keyValues[0] = 109;
        keyValues[1] = 72;
        keyValues[2] = 80;
        keyValues[3] = 13;
        key = 0;
        counter0.begin(COUNTER0_RESOLUTION, COUNTER0_PRESCALER);
        counter1.begin(COUNTER1_RESOLUTION, COUNTER1_PRESCALER);
        run();
    }

    //aktualizace IO a counteru
    void run()
    {
        key = 0;
        //vstupy
        TEMP = inportb(P3);

        //pokud na bitu pro teplo-senzor detekujeme hranu inkrementujeme citac
        if ((TEMP & (1 << TEMP_BIT)) != (lastInput & (1 << TEMP_BIT)))
        {
            tempCounter++;
        }
        //vzdy po sekunde vyresetujeme teplo-citac a spocitame frekvenci
        if (millis() - tempMillis > 1000)
        {
            tempFreq = tempCounter / 2;
            tempCounter = 0;
            tempMillis = millis();
        }
        doorClosed = !(TEMP & (1 << DOOR_BIT)); //spinac dvirek

        //vyhodnocení stisku tlařítka
        if (!(TEMP & (1 << KEY_BIT)))
        {
            if (millis() - keyMillis[index] > keyInterval[index])
            {
                key = keyValues[index];
                //změna periody při více stisků za sebou
                if (keyLastState[index])
```

```
                {
                    keyInterval[index] = 10;
                }
                else
                {
                    keyInterval[index] = 500;
                }
                keyMillis[index] = millis();
            }
        }

        //uložení stavů pro vyhodnocení při dalším cyklu
        keyLastState[index] = !(TEMP & (1 << KEY_BIT));
        lastInput = TEMP;

        //inkrementujeme stav indexu
        index++;
        if (index > 3)
        {
            index = 0;
        }

        //vypis znaku na displey
        TEMP = 0xE0;
        TEMP |= charToByte(string[index]);
        outportb(P1, TEMP);

        //ovladani vystupu
        TEMP = 0xFF;
        TEMP &= ~(((millis() -
 lockMillis > 100) ? 0 : 1) << LOCK_BIT);                //ovladani elektromagnetu
        TEMP &= ~((buzzer ? counter1.getCount() : 0) << BUZZER_BIT);                    //ov
ladani pipaku
        TEMP &= ~(solvePWM(counter0.getCount(), getTurnTablePWM()) << TURN_TABLE_BIT); //ov
ladani talire
        TEMP &= ~(solvePWM(counter0.getCount(), getBulbPWM()) << BULB_BIT);            //ov
ladani zarovky
        TEMP &= ~(128 >> index);                                                        //na
staveni pozice
        outportb(P2, TEMP);

        //updatovani stavu citacu, vzdy na konci cyklu
        counter0.run();
        counter1.run();
    }

    //jsou dvířka zavřená?
    char getDoorClosed()
    {
        return doorClosed;
    }

    //otevře dvířka
    void unlock()
    {
        if (getDoorClosed())
        {
            lockMillis = millis();
        }
    }

    //nastavi PWM hodnotu talire
```

```
    void setTurnTablePWM(int PWM)
    {
        turnTablePWM = (PWM > COUNTER0_RESOLUTION) ? COUNTER0_RESOLUTION : PWM;
    }

    //vrati PWM hodnotu talire
    int getTurnTablePWM()
    {
        return turnTablePWM;
    }

    //nastavi PWM hodnotu zarovky
    void setBulbPWM(int PWM)
    {
        bulbPWM = (PWM > COUNTER0_RESOLUTION) ? COUNTER0_RESOLUTION : PWM;
    }

    //vrati PWM hodnotu talire
    int getBulbPWM()
    {
        return bulbPWM;
    }

    //zapne/vypne pipak
    void setBuzzer(char value)
    {
        buzzer = value;
    }

    //vrati frekvenci naseho uzasneho pokrmu
    //nenasel jsem prevodni graf ani prevodni tebulku
    //pro dané čidlo, tak jsem si udělal jednoduchou
    //rovnici která převádí frekvenci na teplotu
    //nic přesnýho, ale lepší něž nic :)
    int getTempFreq()
    {
        return (0.032 * tempFreq - 156);
    }

    //nastavení pointeru na stringu
    void setStringAddress(char *s)
    {
        string = s;
    }

    //vrátí stisknuté tlačítko
    //pokud není nic stisknuto vrátí 0
    char getKey()
    {
        return key;
    }
};

//13-ENTER
//72-UP
//80-DOWN
//109-MODE
int main(void)
{
    //inicializace proměnných
    char displayString[] = "TIME";
    char stav = 0;
```

```
int runTime = 0;
int powerMode = 0;
int speed = 0;
int pressedKey;
unsigned long stopMillis;
unsigned long displayMillis = 0;
MikroVlnka mikroVlnka;
//do třídy pošle pointer na stringu
mikroVlnka.setStringAddress(displayString);
mikroVlnka.begin();
printf("Cas ohrevu je %i sekund", runTime);
while (1)
{
    //načte stisklé tlačítko z mikrovlnky nebo z klávesnice
    pressedKey = mikroVlnka.getKey();
    if (kbhit())
    {
        pressedKey = getch();
    }

    switch (stav)
    {
    //Zadavani casu pomoci tlacitek "up" a "down", dvirka jsou odemknuta
    //Pri stisku SET se prepne do stavu 1
    case 0:
        switch (pressedKey)
        {
        //stisk šipky nahoru
        case 72:
            //inkrementace doby ohřevu o 10 sekund
            runTime += 10;
            printf("\r                                    \r");
            printf("Cas ohrevu je %i sekund", runTime);
            strcpy(displayString, "    ");
            sprintf(displayString, "%d", runTime);
            break;
        //stisk šipky dolu
        case 80:
            //dekrementace doby ohřevu o 10 sekund
            if (runTime >= 10)
            {
                runTime -= 10;
                printf("\r                                    \r");
                printf("Cas ohrevu je %i sekund", runTime);
                strcpy(displayString, "    ");
                sprintf(displayString, "%d", runTime);
            }
            break;
        //stisk tlačítka SET nebo klávesy ENTER
        case 13:
            //přechod na další stav
            if (runTime > 0)
            {
                strcpy(displayString, "VYKZ");
                printf("\n\rVykon ohrevu je %i", powerMode);
                stav = 1;
            }
            break;
        //stisk tlačítka MODE nebo klávesy m
        case 109:
            mikroVlnka.unlock();
            break;
```

```
        }
        break;
    //Zadavani urovne ohrevu pomoci tlacitek "up" a "down", dvirka jsou odemknuta
    //Pri stisku SET se prepne do stavu stavu 2
    case 1:
        switch (pressedKey)
        {
        //stisk šipky nahoru
        case 72:
            //inkremetace výkonu ohřevu
            if (powerMode < 10)
            {
                powerMode++;
                printf("\r                                    \r");
                printf("Vykon ohrevu je %i", powerMode);
                strcpy(displayString, "    ");
                sprintf(displayString, "%d", powerMode);
            }
            break;
        //stisk šipky dolu
        case 80:
            //dekrementace výkonu ohřevu
            if (powerMode > 0)
            {
                powerMode--;
                printf("\r                                    \r");
                printf("Vykon ohrevu je %i", powerMode);
                strcpy(displayString, "    ");
                sprintf(displayString, "%d", powerMode);
            }
            break;
        //stisk tlačítka SET nebo klávesy ENTER
        case 13:
            //přechod na další stav
            if (runTime > 0)
            {
                strcpy(displayString, "VYKM");
                printf("\n\rVykon motoru je %i", speed);
                stav = 2;
            }
            break;
        //stisk tlačítka MODE nebo klávesy m
        case 109:
            mikroVlnka.unlock();
            break;
        }
        break;

    //Zadavani urovne motoru pomoci tlacitek "up" a "down", dvirka jsou odemknuta
    //Pri stisku SET se prepne do stavu stavu 3
    case 2:
        switch (pressedKey)
        {
        //stisk šipky nahoru
        case 72:
            //inkremtace výkonu motoru
            if (speed < 10)
            {
                speed++;
                printf("\r                                    \r");
                printf("Vykon motoru je %i", speed);
                strcpy(displayString, "    ");
```

```
                    sprintf(displayString, "%d", speed);
                }
                break;
            //stisk šipky dolů
            case 80:
                //dekrementace výkonu motoru
                if (speed > 0)
                {
                    speed--;
                    printf("\r                                    \r");
                    printf("Vykon motoru je %i", speed);
                    strcpy(displayString, "    ");
                    sprintf(displayString, "%d", speed);
                }
                break;
            //stisk tlačítka SET nebo klávesy ENTER
            case 13:
                //přechod na stav ohřevu
                if (runTime > 0 && mikroVlnka.getDoorClosed())
                {
                    mikroVlnka.setTurnTablePWM(speed * 10);
                    mikroVlnka.setBulbPWM(powerMode * 10);
                    stopMillis = millis() + (unsigned long)runTime * 1000;
                    displayMillis = 0;
                    stav = 3;
                    printf("\n\r");
                }
                break;
            //stisk tlačítka MODE nebo klávesy m
            case 109:
                mikroVlnka.unlock();
                break;
            }
            break;
        //ohrev
        //odpocitava sekundy do konce ohrevu, pri uplynuti casu odemkne dvirka a vrati se d
o stavu 0
        //pri stisku SET vypne ohrev, odemkne dvirka a vrati se do stavu 0
        case 3:
            switch (pressedKey)
            {
            case 72:
                break;
            case 80:
                break;
            //stisk tlačítka SET nebo klávesy ENTER
            //Trochu ošklivě ukončí ohřev
            case 13:
                stopMillis = 0;
            }
            //každých 100 ms aktualitujeme počet sekund
            //do konce ohřevu a "teplotu" pokrmu
            if (millis() - displayMillis > 100)
            {
                printf("\r
   \r");
                printf("Zbyva %i sekund do vypnuti ohrevu a teplota pokrmu je ", (stopMilli
s - millis()) / 1000, mikroVlnka.getTempFreq());
                printf("%i \370C", mikroVlnka.getTempFreq());
                strcpy(displayString, "    ");
                //při sudém počtu sekund do konce ohřevu zobrazujeme teplotu
                //a při lichém konec sekund do konce ohřevu
```

```
                sprintf(displayString, "%d", ((stopMillis -
 millis()) / 1000) % 2 ? (stopMillis - millis()) / 1000 : mikroVlnka.getTempFreq());
                displayMillis = millis();
            }
            if (millis() > stopMillis)
            {
                //konec ohřevu
                //zapnutí bzučáku
                //otevření dvířek
                //vypnutí motoru a žárovky
                printf("\n\rPane, vase jidlo je hotovo a pripraveno k vyhozeni do kose :) p
reji dobrou chut\n\r");
                strcpy(displayString, "DONE");
                mikroVlnka.setTurnTablePWM(0);
                mikroVlnka.setBulbPWM(0);
                mikroVlnka.unlock();
                mikroVlnka.setBuzzer(1);
                //bzučák bude bzučet 1 sekundu
                stopMillis = millis() + 1000;
                stav = 4;
            }
            break;
        case 4:
            if (millis() > stopMillis)
            {
                //vypnutí bzučáku a návrat na stav 0
                mikroVlnka.setBuzzer(0);
                printf("\n\rCas ohrevu je %i sekund", runTime);
                strcpy(displayString, "TEMP");
                stav = 0;
            }
            break;
        }

        //escape tlačítko zastaví běh programu
        if (pressedKey == 27)
        {
            mikroVlnka.setTurnTablePWM(0);
            mikroVlnka.setBulbPWM(0);
            break;
        }

        mikroVlnka.run();
    }
    //konec programu a návrat do Borlanda
    while (!kbhit())
        ;
    return 0;
}
```