Stand 16.06.08

Projekt Thermen-Steuerung

von Manuel Schreiner Mat.-Nr.: 713586
Betreuer: Prof. Rückle

Projekt-Kurzbeschreibung:

Ziel des Projektes ist es ein Gerät zu entwickeln, welches die Temperatur einer Gas-Therme einstellen kann, ohne bauliche Maßnahmen an der Therme vor zu nehmen. Je nach Tageszeit soll die Therme auf einen Nachtbetrieb fahren können oder tagsüber einen Sollwert (1-8) halten. Zur Verfügung steht an der Therme nur ein Drehknopf mit den Sollwerten 1 bis 8. Das Gerät soll unabhängig funktionieren und ein Display mit Bedienelementen besitzen um Einstellungen vor zu nehmen. Zusätzlich soll eine Aktivierung/Deaktivierung der Therme über ein Handy möglich sein.

Inhaltsverzeichnis

1. Aufgabenstellung

1.1 Anforderungen
Die Regelung des Drehknopf übernimmt ein Modellbauservo, das einen Winkel von 0 – 240° fahren kann. Dazu muss eine Ansteuerung des Servos programmiert werden. Es sollen hierbei mindestens 8 Werte möglich sein.
Für die Nacht-/Tag-Programmierung muss ein passender Uhrenbaustein ausgewählt und implementiert werden. Die Einstellung erfolgt mittels Display. Des Weiteren soll eine Fern-Steuerung mittels Handy (Siemens) möglich sein. Wünschenswert wäre die Steuerung via SMS, Bedingung ist das Schalten via Anruf. Für die Programmierung wird WinAVR in Verbindung mit GCC eingesetzt. Als Prozessor dient ein AVR RISC Mega.

1.2 Anforderungsliste

Pos.	Wunsch/Forderung	Bezeichnung	evtl. vorgeschriebener Wert
1	Forderung	Drehknopfbedienung mittels Servo	
2	Forderung	Minimale Positionen des Servos	8 (Wunsch 16)
3	Forderung	Statusdisplay mit Bedienung	(Display mit Joystick)
4	Forderung	Zeitprogammierung (Uhr)	(Wunsch Handy-Uhr auslesen)
5	Forderung	Steuerung mittels Handy	per Anruf (Wunsch SMS)
6	Forderung	Hardware: AVR RISC	ATMEL ATMEGA 8535
7	Forderung	Progr.-Sprache: AVR-C	
8	Wunsch	Batteriebetrieb	
9	Forderung	Manuelle Bedienung	
10	Forderung	Automatischer Timer Betrieb	

2. Planung

2.1 Hardware

Benutzt wird ein ATMEL Butterfly Board mit einem MEGA 169 Mikrocontroller mit einem internen Takt von 8MHz, der soweit alle Funktionen des Projektes übernehmen kann und über genügend IO-Ports für die Peripherie verfügt. Als Schnittstelle zum Handy dient eine RS232 Schnittstelle mit Pegelwandler. Das Handy dient sowohl zur Remotebedienung als auch als Uhrenbaustein, der die aktuelle Uhrzeit enthält. Die Uhrzeit kann dann per Befehlssatz ausgelesen werden und aktualisiert die interne Zeit des Mikrocontrollers. Um Energie zu sparen kann das Servo im späteren Betrieb abgeschaltet werden. Als Einstellmöglichkeit dient ein Display mit 5 Knöpfen (Joystick) für die Menüführung.

2.2 Software

Die Software besteht aus folgenden Modulen:

2.2.1 UART Programmierung

- Die serielle Schnittstelle dient zur Datenübertragung zum Handy. Benötigt wird hierzu ein Empfangsbuffer, der bei einem Linebrake den gespeicherten Buffer auswertet. Theoretisch ist ein Sendebuffer empfehlenswert, dieser wird jedoch wegen geringer Komplexität des Gesamtaufbaus vernachlässigt und bei jedem Senden der Ausgabe direkt verarbeitet.
-

2.2.2 Befehlssätze für Siemens C35 Implementierung

- Die Befehlssätze des Siemenshandys basieren auf dem GSM AT Befehlssatz. Die wichtigsten Befehle in dem Projekt sind Erkennung eines Anrufes, ermitteln der Rufnummer des Anrufers, auflegen und auslesen der Zeit im Handy.
-

2.2.3 Servo-Steuerung & Timing

- Ein Modellbauservo ist ein Modul, das seinen Drehwinkel je nach Modell von 0 bis 240° drehen kann. Ausschlaggebend für die Position ist ein Signal, das eine Flanke von 1ms bis 2ms erzeugt, gefolgt von einer Ruhephase von mehr als 10ms. (1ms = 0° - 2ms = 240°). Da bei der Steuerung der Heizung keine hohe Auflösung des Servos gefordert ist, wird in diesem Projekt nur eine Aufteilung von 16 Schritten vorgenommen.
-

2.2.4 Display-Implementierung

- Das Butterfly Board bringt ein LCD mit sich, das durch einen internen LCD Controller des Mega 169 angesteuert wird. Durch eine angepasste Version des Atmel Treibers für das LCD ist es auch unter AVR-GCC zu verwenden.
-

2.2.5 Ansteuerung der Tasten

- Das Ansteuern der Tasten geschieht mittels Interrupt. Dieser stellt einen „char key“ bereit, in dem notiert ist, welcher Button gerade gedrückt wird. Sollten die Taster prellen, ist eine Softwareentprellung sinnvoll.
-

2.2.6 Menüführung

- Es muss die Möglichkeit geben sowohl einen manuellen als auch einen automatischen Modus auszuwählen und eine Art Zeitschaltuhr zu programmieren.
-

2.2.7 Timetables

- Die Timetables für die unterschiedlichen Timer werden im internen EEPROM des Controllers abgelegt. Nach einem möglichen Spannungsverlust bleiben die Daten im Speicher erhalten.

3. Realisierung

3.1 Hardware

Das Projekt besteht aus insgesamt 4 Modulen:

- Modellbauservo: Zum Ansteuern des Reglers an der Gastherme
- Mobiltelefon: Zum Fernsteuern der Heizung und für die Uhrzeit
- Display- und Controls-Einheit: ATMEL Butterfly
- Netzteil für Handy und Butterly

3.1.1 Pin- & Portbelegung

Belegung der einzelnen Ports des Butterflys:

Joystick hoch:	PINB6
Joystick runter:	PINB7
Joystick links:	PINE2
Joystick rechts:	PINE3
Joystick mitte:	PINB4
Servo:	PINB0
Temperatursensor:	ADC0

3.2 Software

3.2.1 UART Programmierung

Die UART Schnittstelle dient zur Kommunikation mit dem Mobiltelefon. Diese besteht aus einem Sende- und Empfangsteil. Das Sendeteil kennt mehrere Modi: ein Byte senden, einen String senden, einen String mit Zeilenumsprung senden. Für die jeweiligen Modi wurde ein(e) void/function vorgesehen:

- int uart_putchar(char c): Sendet einen Char über die serielle Schnittstelle.
- void print(const char *s): Sendet eine String über die serielle Schnittstelle.
- println(const char *s): Sendet einen String mit Zeilenumsprung über die ser. Schnittstelle
- void print_int(int i): Gibt einen Integer als String auf der ser. Schnittstelle aus.

Die ISR „SIGNAL(SIG_UART_RECV)“ empfängt Zeichen von der seriellen Schnittstelle und speichert diese in einem Puffer. Wird das Zeichen 10 oder 13 empfangen, wertet die Prozedur void find_command(const char *s) den Puffer aus.
Folgende Kommandos werden dabei verstanden:
„+CCLK:“ gefolgt von der Uhrzeit, die das Handy ausgibt.
„+CLCC:“ gefolgt von der Rufnummer des momentanen Anrufers.
„RING“ für einen eingehenden Anruf.

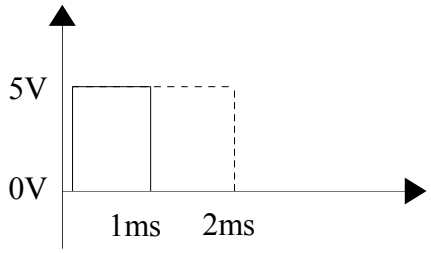
3.2.2 Befehlssätze für Siemens C35 Implementierung

Einstellungen der ser. Schnittstelle: 19200,N,1
Register des Mikrocontrollers: UBRRH = 0, UBRRL = 25, UCSRB = 0b10011000

Befehl: AT^SCLK? (Uhrzeit auslesen)
Antwort: +CCLK: <time>
<time>: string type value; format is "yy/MM/dd,hh:mm:ss",

Befehl: AT+CLCC (Anrufer erfragen)
Antwort: +CLCC: <id1>,<dir>,<stat>,<mode>,<mpty>,<number>,<type>
Antwort: „RING“
Eingehender Anruf.

3.3.3 Servoansteuerung



Die Stellinformation wird dem Servo mitgeteilt mit Hilfe einer Flanke von 1-2ms gefolgt von einer Ruhephase zwischen 10 und 20 ms. Entscheidend ist jedoch die Flanke. Die Servosteuerung wird über die Timer ISR SIGNAL(SIG_OVERFLOW0) erledigt. Diese wird etwa alle 100µs aufgerufen. Das Servo kann nach Definition grob in einem Bereich von einer Flanke von 1ms bis 2ms gesteuert werden. D.h. 1ms kann wie ein zeitlicher Offset angesehen werden, zudem ein Arbeitsbereich von 0...1ms folgt. Durch Tests ergab sich ein Offset von 9 Intervallen (0,9ms) mit einem Arbeitsbereich, der zwischen 0 und 15 Intervallen liegt (0...1,5ms). Um die Bereiche des Signals in Offset-Bereich, Arbeits-Bereich und Ruhe-Bereich zu unterteilen, wurden Offset- und Arbeits-Bereich zu einem „On-Bereich“ erweitert. Eine Variable servotmp wird pro Iterationsinterval incrementiert. Ist diese unter Offset + Servoposition, wird der Port an dem das Servo hängt auf High gestellt. Liegt servotmp über Offset + Servoposition ist der Port Low bis servotmp 200 erreicht. Ab 200 wird servotmp auf null gesetzt und die Variable setservo incrementiert. setservo hat den Auftrag nach 100 Impulsen, das Servo ab zu stellen, d.h. das Ausgeben der Impulse zu steuern. Empfängt ein Servo keine Impulse, geht es in einen stromspar Modus.

- **3.2.4 Display-Implementierung**
- Durch den vorgegebenen Treiber von ATMEL, wird an der LCD Implementierung nicht mehr viel programmiert. Die gewünschte Ausgabe wird per String an das LCD Modul übergeben und dann als Laufschrift ausgegeben.

- **3.2.5 Ansteuerung der Tasten**
- Ändert sich eine Taste, so werden die Tasten per Interrupt eingelesen, zwischengespeichert und eine Countervariable gesetzt, welche zur Endprellung genutzt wird. Generell sollte die Tasteneingabe in Realzeit erfolgen, d.h. für einen Menschen ist 1ms nahe zu Realzeit. Die Countervariable wird daher auf 10 gesetzt und im Timer für die Servoansteuerung, welcher alle 100µs aufgerufen wird, heruntergezählt.
- Erreicht der Counter null, wird der Wert der zwischengespeicherten Taste auf die Variable „char key“ übertragen und steht dem Rest des Systems zur Verfügung. Dieses darf die Variable key ausschließlich lesen, da es sonst zu Datenverlusten kommen könnte. Die Ausgabe von „key“ orientiert sich an einem Nummernpad einer Tastatur:
- hoch 8, runter 2, links 4, rechts 6, center 5.
-

3.2.6 Menüführung

Siehe Anhang „Bedienungsanleitung“.

3.2.7 Time-Tables

Um die Schaltzeiten der Therme zu speichern wird ein nichtflüchtiger Speicher benötigt, den der verwendete Mikrocontroller in Form eines EEPROMs mit sich bringt. Das EEPROM ist ein Blockspeicher, ähnlich wie der RAM des Controllers. Um die Daten sinnvoll und platzsparend unter zu bringen wurde folgende Struktur von Blöcken verwendet:

Byte	**Bit7**	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
1 Tage	-	So	Sa	Fr	Do	Mi	Di	Mo
2 Stunde	-	-	-	x	x	x	x	x
3 Minute	-	-	x	x	x	x	x	x
4 Heizung	-	-	-	-	x	x	x	x

Tabelle 3.2.7a

Ein Block im EEPROM besteht dabei aus 4 Bytes. Das erste Byte in jedem Block ist zuständig für die Wochentage. Wie in Tabelle *3.2.7a* zu erkennen ist, werden die einzelnen Bits für die verschiedenen Tage gesetzt. Somit kann mit einem Timetable eine Programmierung für mehrere Tage erfolgen (wie z.B. Wochentags, Wochenende oder Individuell). Die folgenden Bytes verwalten Stunde und Minute, sowie die Stellung des Reglers an der Heizung. Sind Uhrzeit und Zeitstamp der Timetable gleich, wird der eingespeicherte Wert für die Stellung des Reglers übergeben.

10. Fazit

Weitgehend wurden die erwünschten Funktionen umgesetzt: Die Therme ist sowohl in einem manuellen Modus, als auch in einem automatischen Modus zu bedienen. Durch die integrierte Uhr im Mobiltelefon, kann auf ein Uhrmodul verzichtet werden, die Programmierung der Uhrzeit geschieht im Mobiltelefon selbst. Um auf die verschiedenen Wochentage eingehen zu können, musste lediglich der Wochentag im Hauptmodul erfragt werden. Es können Zeiten für verschiedene Tage eigegeben werden, zu denen die Steuerung den Regelknopf der Therme verändert.
Als Verbesserung wäre eine Remotesteuerung via SMS wünschenswert gewesen oder bei Fernsteuern über einen eingehenden Anruf, die Eingabe von Nummern, die fernsteuern dürfen. Letztendlich stand jedoch das zeitliche Steuern der Therme im Vordergrund, welches vollständig umgesetzt wurde.
Wegen Zeitmangel musste jedoch auf gewissen Komfort verzichtet werden, der sich im kleinen Detail ersichtlich macht: Die eingesetzte Hardware ist nicht ganz optimal, da das ATMEL Butterfly Board ein Entwicklungsboard ist, das zwar viele Vorzüge mit sich bringt, aber auch viele Komponenten somit von Anfang an mitbestimmt. Vorteile ergaben sich dadurch durch schnellere Entwicklung und gute Dokumentation Seitens ATMELs. Als Display wäre evtl. ein mehrzeiliges Display besser gewesen, das auch mehr Zeichen pro Zeile zugelassen hätte.