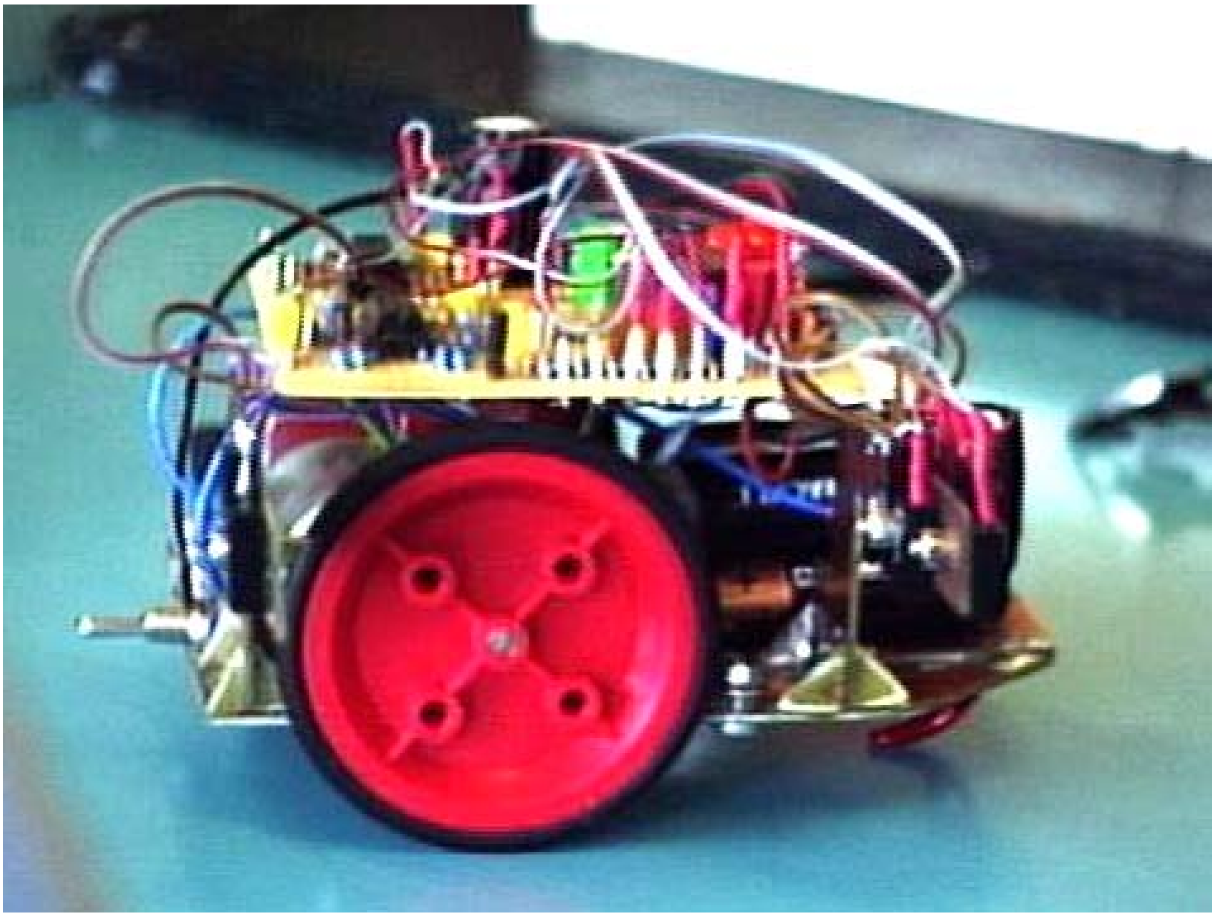
Group Project Report
MICROMOUSE

Microprocessor Controlled Vehicle



Name:	Michael Gims,	9801352
	Sonja Lenz,	9801350
	Dirk Becker	9801351
Course:	BEng 2	
Group:	B2	
Date:	05/May/1999	

UNIVERSITY of EAST LONDON

1. Contents

2. Introduction

The UEL Group Project should force students working together as a small group. Therefore a vehicle, called “Micromouse” was to be designed and built up.
Aim of the project is to win the final race, which consists of three rounds each 2 laps on a small rectangular maze.
The micromouse project is sponsored by the IEEE.

3. The Maze and the Race

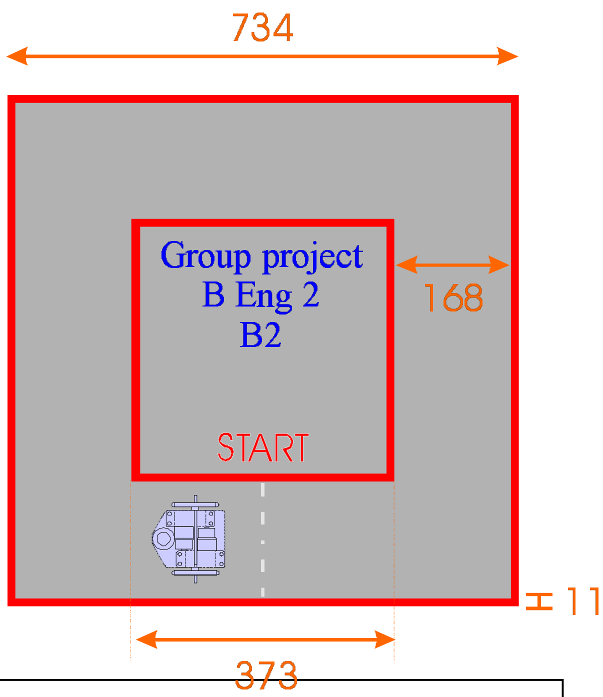


figure 1 – The Maze

The above figure shows the maze, in which the micromouse has to do two laps as fast as possible. The start an finish is in the middle of one side. The micromouse can be adjusted to the walls at the beginning of the race. The micromouse should run with minimum wall contact and has to find its way itself. Therefore in the most cases sensors are used, which can detect the walls.

4. Constructional ideas

Two complete different ways of designing the micromouse are possible. One is the conventional analogue method with comparators and analogue control of the motors. The second, the digital way can either be realised by using digital logic devices or by using a microprocessor. The opportunity of using a microprocessor is the easy and convenient adjustment of the software for different tasks and to increase the overall speed without any change of hardware or the trial and error method with a lot of variable resistors.
Therefore we used a Microchip PIC microcontroller, which is very fast in operation and because of its reduced instruction set (RISC processor with Harvard architecture) easy to handle. The software and the documentation are free downloadable on the Microchip's website (Http//www.microchip.com).
The Harvard architecture allows execution of one instruction in only one processor cycle, because the data and the address bus are completely separated.
Also I/P and O/P of the PIC are very easy to use, because they can be directly accessed by the program.
The motor control is realised with two half bridges, which contain each 2 logic level MOSFETs one for running and one for braking. These Transistors can be directly driven by CMOS-Logic devises and are capable of high voltages and high current. Driving backwards was thought not to be necessary during the design process. At the end the lack of this possibility was remarkable.

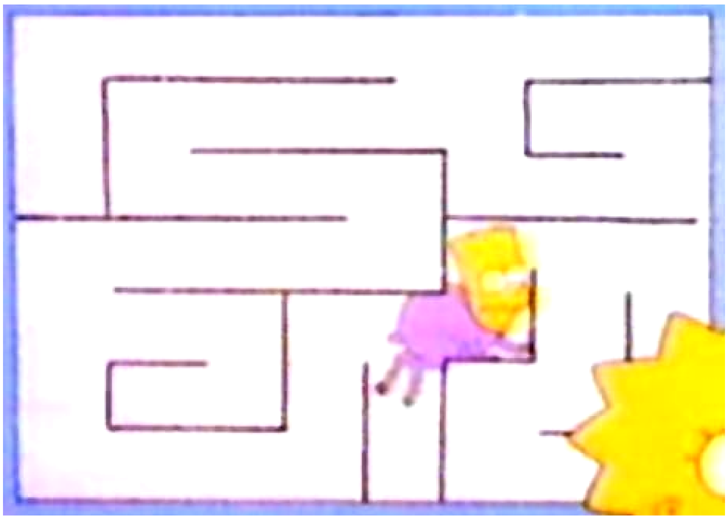


figure 2 – Not only mice go through a maze

For wall detecting pulsed infrared sensors are used. The pulsing of the sensors increases their working distance enormous and makes variable resistors unnecessary. With these sensors it is also possible to read out wheel reflectors, but the software for them could not be finished in time.
We tried to built up the micromouse as compact and light as possible for having sufficient space for turning and running the micromouse between the walls of the maze.
Reduction of weight results in a better mobility.

5. Mechanical Construction

From the figure below it can be seen that the gearboxes are mounted as close as possible to reduce the size of the chassis.

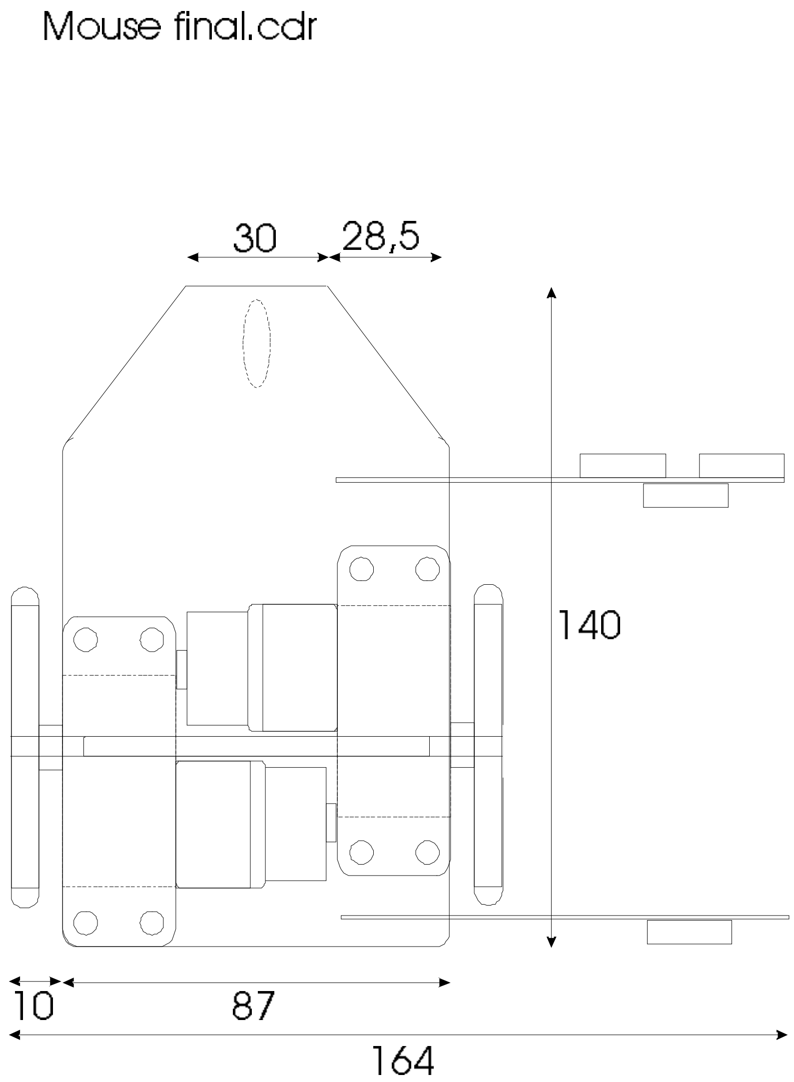


figure 3 - Mouse layout - topview

Everything on the chassis was mounted in a way that we could get as close as possible to the maze ground to get the centre of gravity down.
For minimum weight the sensor arms are build out of epoxy (the same material as the chassis) and soldered to the ground plate. The ball-bearing we used first was replaced by a cloth hanger. This also led to better steering of the micromouse because of the reduced weight in the front.

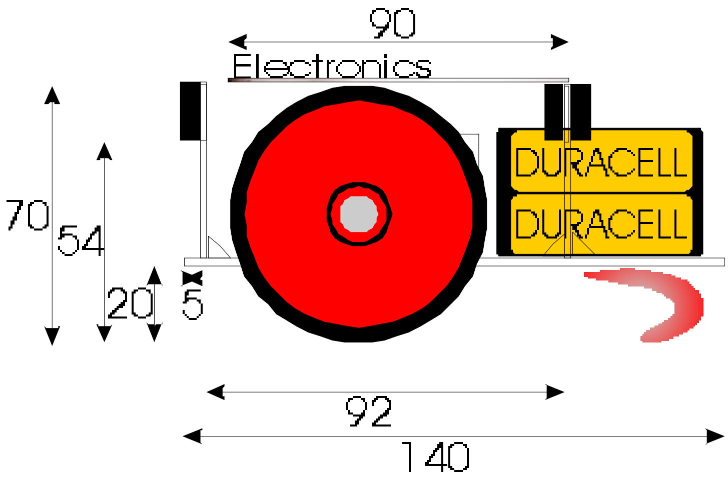


figure 4 - Sideview

As it can be seen in the schematic drawing above, the main circuit board is placed directly above the gear boxes and fixed with a long screw and the sensor arms. To prevent bouncing during the acceleration process the batteries where put along to the front of the vehicle.
The distance between the two sensor arms where chosen that the micromouse can turn when the last sensor just leaves the wall. At the end we found out that this leads to a too long braking distance. The sensors should be more in the front of the micromouse in accordance to the delay through the braking process.
The sensors were located to look on the top of the wall. Side-looking sensors with exact distance measurement could result in a very proper function.

6. Electrical Construction

a) The Sensors

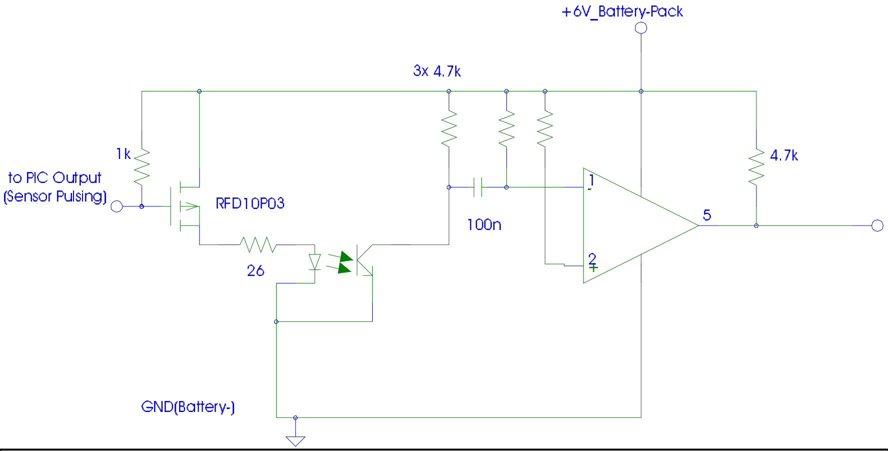


figure 5 – Schematics of one Sensor-stage

The figure above shows the circuit diagram of one of the sensors. The IR-Diode is pulsed by the MOSFET. When the IR-Transistor detects reflected light (which means a wall), the input of the comparator LM339 is pulled to ground and at the output of the comparator occurs a high impulse. The 100nF coupling capacitor blocks DC and low frequent signals. Therefore ambient light can not interfere.

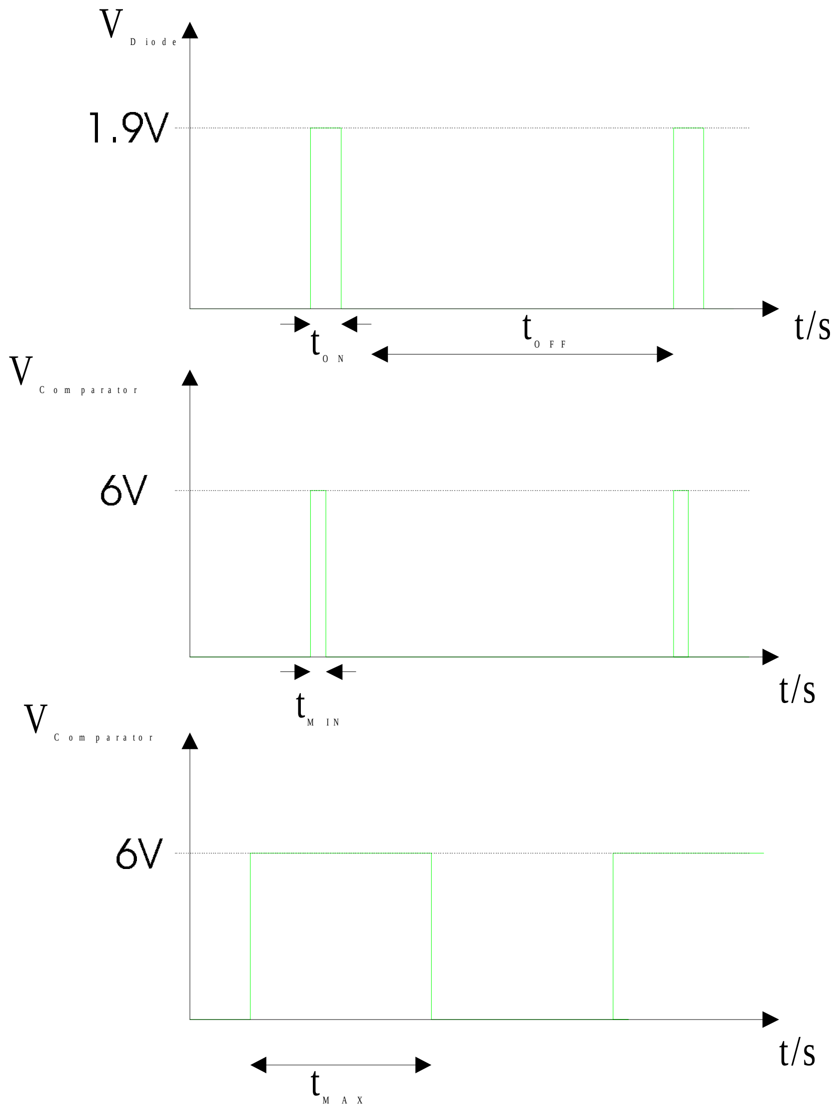


figure 6 - Diode Voltage and Comparator Output

The sensors are switched on for t_{on}=20µs and the corresponding output times differ between t_{min}=1µs and t_{max}=75µs depending on the distance to the reflecting wall. The dead-time of the diodes is more than 1ms, so that the ratio between ON and OFF is more than 1:25, because otherwise the diodes would be destroyed due to the high current. This current is about 150mA, which is more than three times higher than their maximum capability.

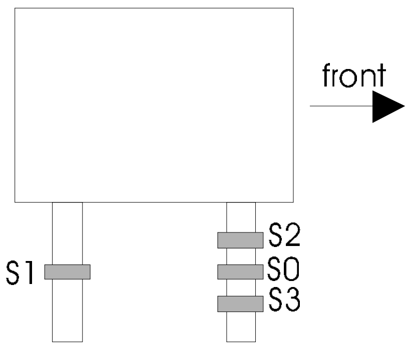


figure 7 – Sensor locations on the mouse

b) The Processor

The PIC microprocessor runs at a speed of 10Mhz with a supply voltage direct from the batteries (3V – 6V).
An additional reset button is connected to the /MCLR pin in order to reset the PIC manually during program operation.
The sensors are connected directly to the input pins.
The operating speed of 10MHz is sufficient to provide a Pulse Width Modulation (PWM) with a frequency of about 4kHz and also control the vehicle and check the sensors.
The outputs drive directly the power stage via a CMOS4001 latch.

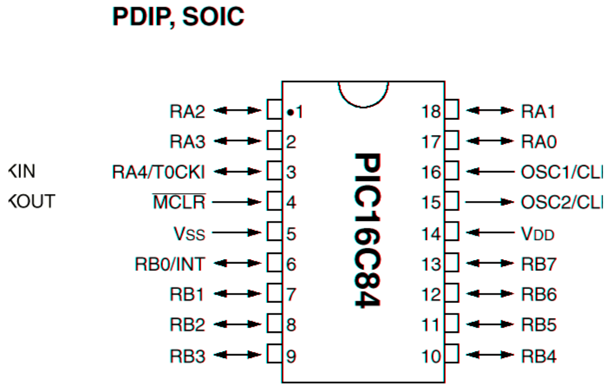


figure 8 - Pinout of the PIC 16x84

The hardware development of the µPC circuit created no difficulties. Only the in-circuit-programming feature could not be realised, because of too less current provided by the simple programmer circuit.

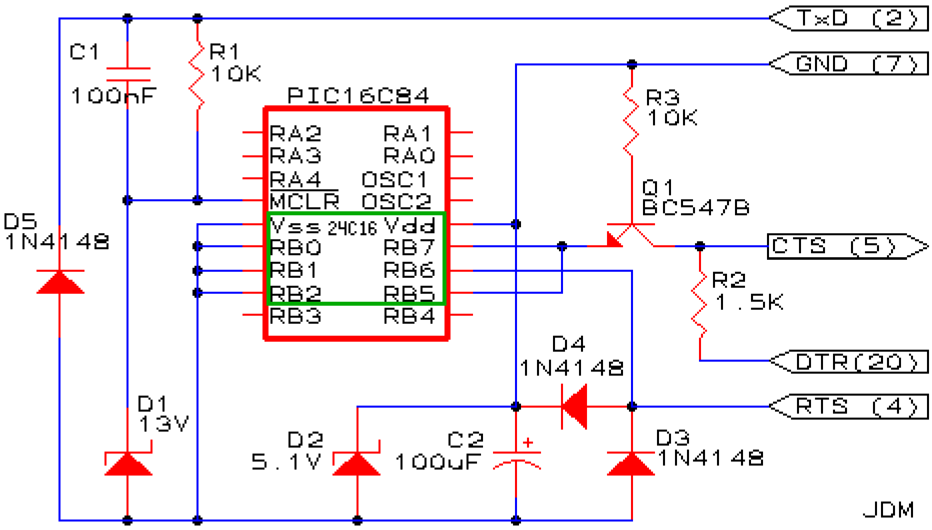


figure 9 - JDM programmer circuit

The used programmer is a simple RS232 one developed by Jens D. Madsen, which does a good job, but causes some trouble with the in-circuit-programming. Hence the PIC had always to be changed between the programmer and the micromouse. The needed software is supplied with the programmer.

c) The Power stage

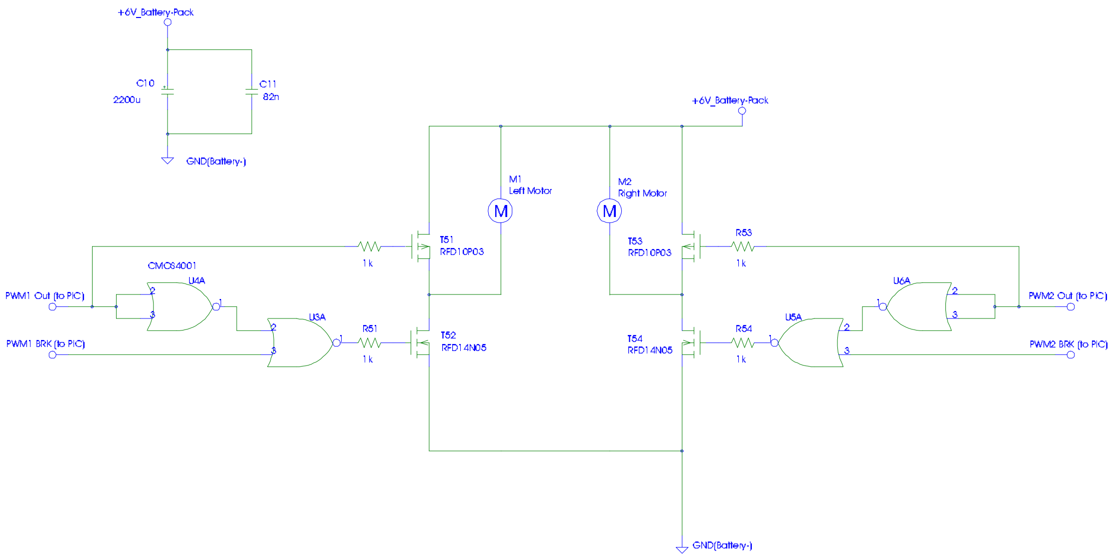


figure 10 – Motor driver stage (power stage)

The motor driver stage was designed only for driving forward and braking by short circuiting them. It consists of two half-bridges, one for each motor. The motors are driven by Harris Semiconductors logic-level MOSFets, which can be switched directly with CMOS logic sources.
The CMOS 4001 latch is used to prevent the MOSFETs from damage, when they are both switched on. This can occur when the P-channel MOSFET has got a low level and the N-channel MOSFET has got a high level at the gate. This could also be done by software, but a hardware protection was preferred.

PWM (from PIC)	BRK (from PIC)	N-Gate	P-Gate	Motor
0	**0**	**0**	**0**	**brake**
0	**1**	**0**	**0**	**brake**
1	**0**	**1**	**1**	**run**
1	**1**	**0**	**1**	**idle**

Boolean equations to prevent the not allowed condition:

$$\mathrm{P-Gate} = \mathrm{BRK}$$

$$\mathrm{N-Gate} = \mathrm{PWM} \cdot \overline{\mathrm{BRK}}$$

$$\mathrm{N-Gate} = \overline{\overline{\mathrm{PWM} \cdot \overline{\mathrm{BRK}}}}$$

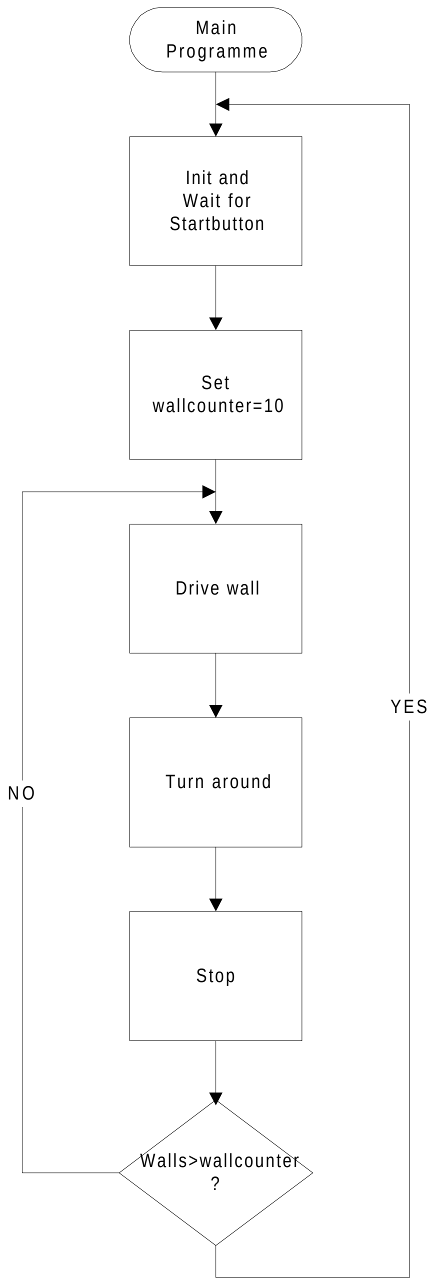
$$\mathrm{N-Gate} = \overline{\overline{\mathrm{PWM}} + \mathrm{BRK}}$$

, which results in the circuit shown in the schematics above.

7. The Software

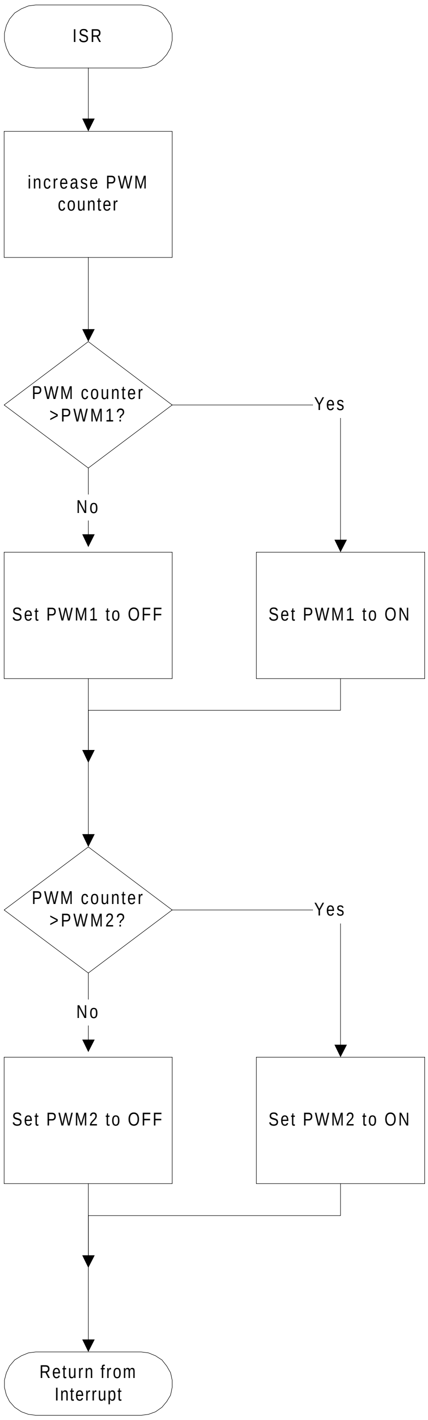
The software for the vehicle must be able to drive the mouse straight ahead and to turn right around the corner. For driving straight ahead the software has to control the motors in an suitable way. Therefore a PWM (Pulse Width Modulation) with a resolution of 16 bits for each wheel was developed.
The program has to check the sensors in order to control the vehicle by detecting the walls.

a) The Main Program



During the initialisation the I/Ps and O/Ps are set. Also all the needed variables are presetted or cleared and the ISR (Interrupt Service Routine) for PWM is started.
Then the program waits for the START button (Input RA2 of the PIC) to be pressed.
After that the PIC drives the vehicle along the wall, until no sensor has wall detection. Then the TURN subroutine is started and the wallcounter decreased by one. The mouse stops and if not all walls are done, the mouse starts with the next straight driving process.

b) The ISR for the PWM



The ISR is called with every timer overflow (every 16µs). At first a helping counter is increased. he ISR checks now, if the desired PWM value is higher or less than the helping counter. If the desired PWM value is less, the appropriate PWM output is set to low and otherwise to high level.
With every call of the ISR the timer has to be presetted again to the 16µs value (230dec.).

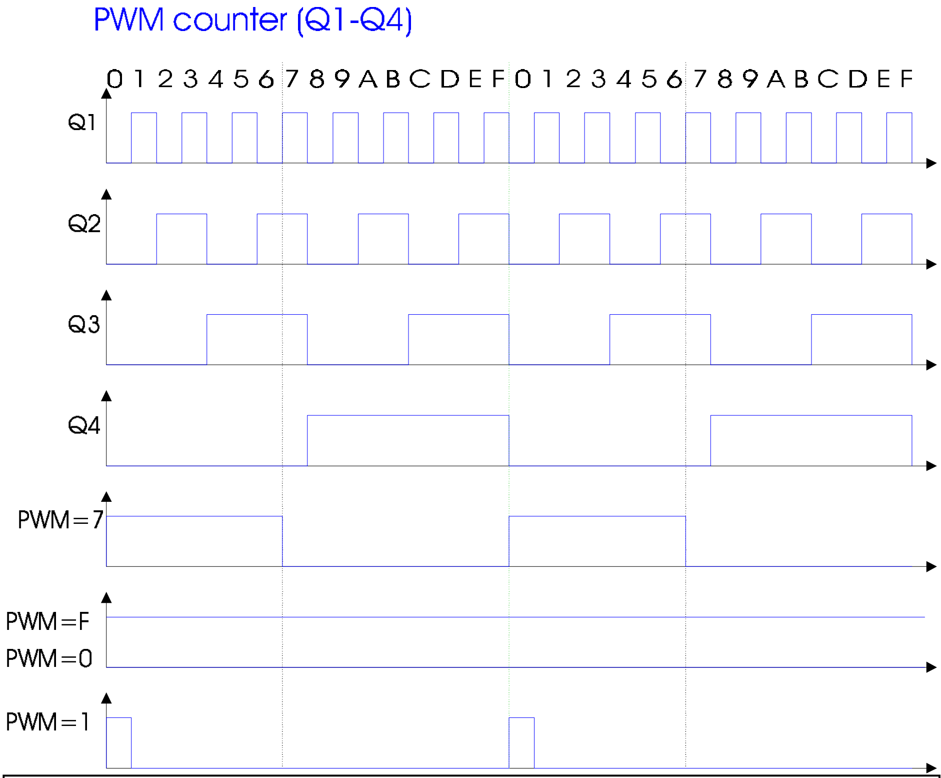
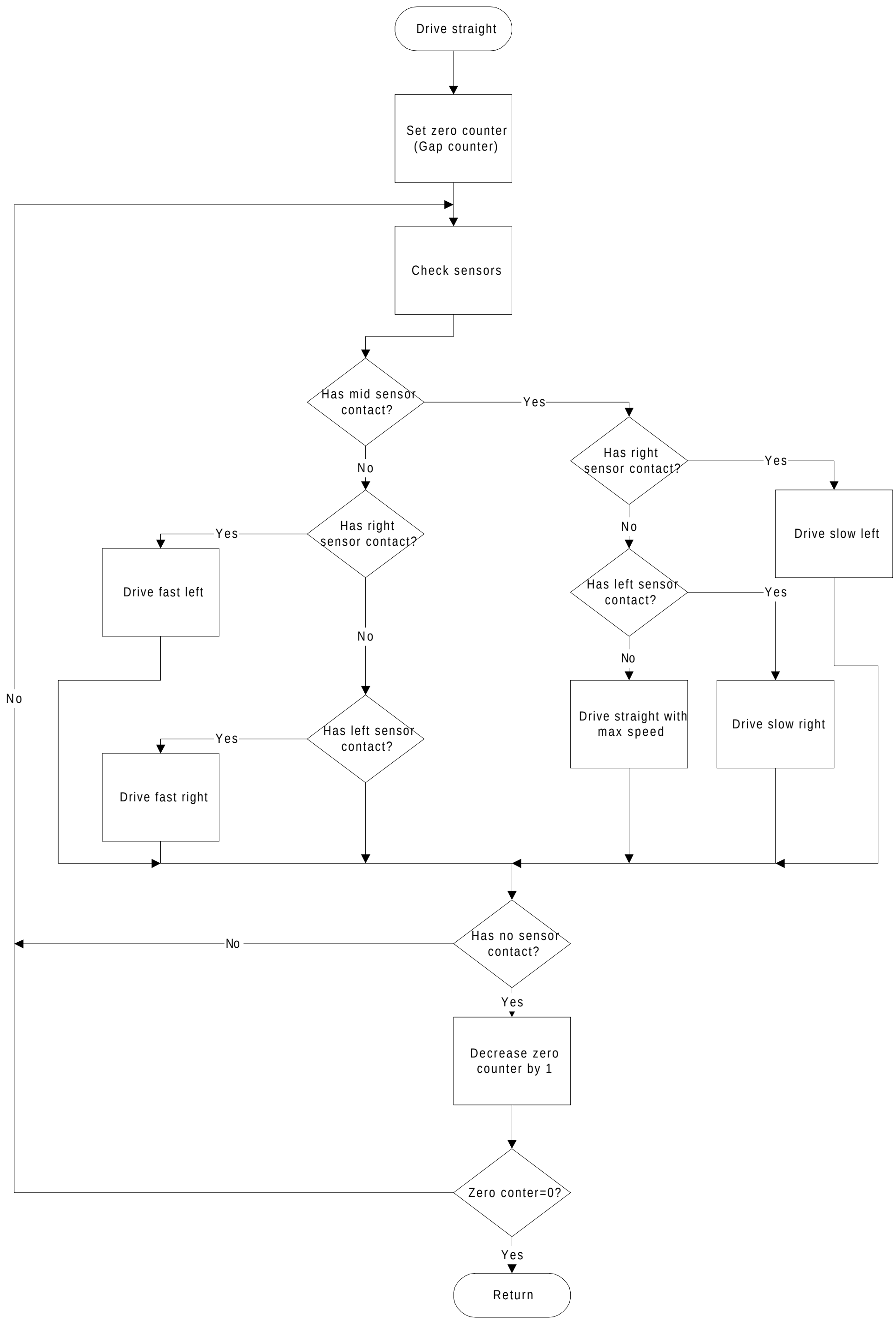


figure 11 – Timing diagram for PWM

The above diagram shows in an elegant way, how the PWM actually works. The bits Q1 .. Q4 represent the corresponding bits of the PWM helping counter. The ISR compares them with the desired PWM and switches the outputs high or low, depending on the result of the comparison.

c) Driving along the wall



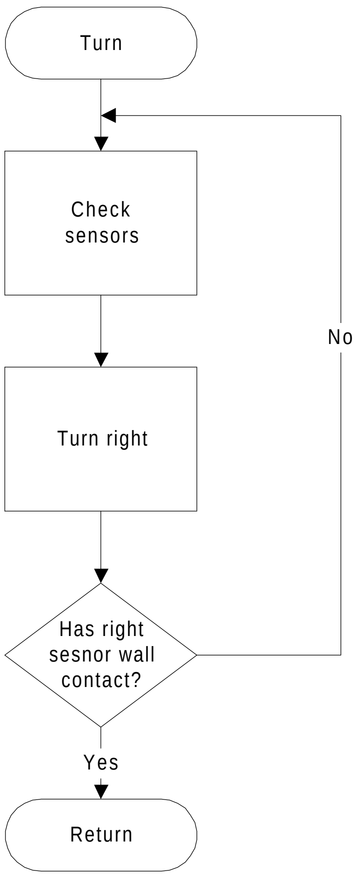
Depending on the sensors, the mouse has either to drive straight on or to steer left or right. The control algorithm is a modified bang-bang mechanism.

There are 5 different stages:

Stage 1:	only mid sensor has contact	⇒	drive ahead
Stage 2:	mid and right sensor has contact	⇒	drive slightly right
Stage 3:	only right sensor has contact	⇒	drive right
Stage 4:	mid and left sensor has contact	⇒	drive slightly left
Stage 5:	only left sensor has contact	⇒	drive left

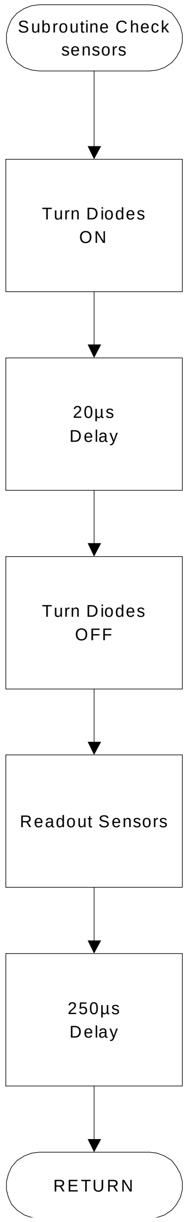
When none of the sensors have contact, a gap counter is decreased and when this counter is equal to zero, the mouse starts the stop and then the turning process.

<u>**d) The turn process**</u>



The turning process is a very simple subroutine, which brakes the right wheel and turns with the left one around, until the first (the right) wall-sensor has contact. Due to the force of inertia and the wheel-slip the vehicle turns a bit further and the as next started straight driving routine adjusts the car again straight towards the wall.

e) Checking the sensors



The diodes are turned on, by setting the appropriate output of the PIC (RA4) to a low level. Then a 20µs delay is performed, to switch the diodes for a sufficient time on. After turning off the diodes again the sensors are read out. To provide a long enough time between two readout tasks, a turnoff delay of 250µs delay is executed as recovering time for the diodes.
The reflected signal has a duration between 1µs and 75µs depending on the distance to the wall.

8. Possible Improvements

The major problem of our Micromouse is the lack of driving backwards, which can be very useful for braking with great efficiency and to cancel out dynamic movements. Also the implementation of the wheel sensors would increase the overall speed dramatically. With the knowledge of the actual position the acceleration, braking and turning processes could be sped up by slowly accelerating then driving with maximum speed and before the end of the wall again reducing the speed. The wheel sensors were tested, but unfortunately there was not enough time to implement them in the software.

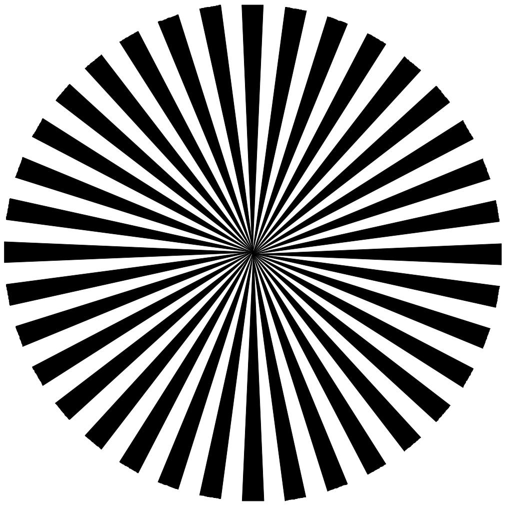


figure 12 – Prototype of a wheel reflector

9. Conclusion

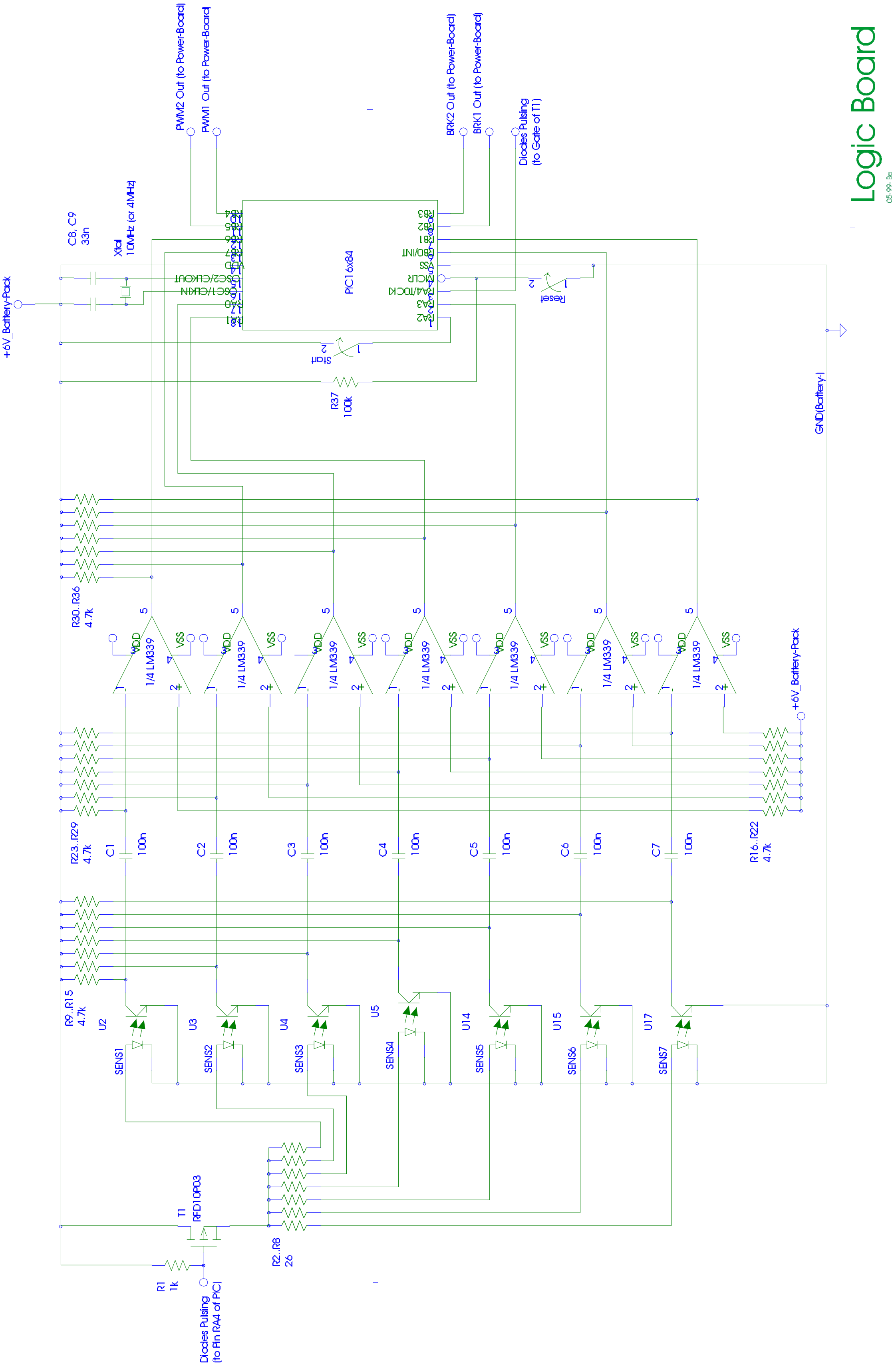
The micromouse-project was a very useful experience for building a practical device in a group. The theory was not put in the foreground – The mouse had to run and there was no question WHY. We all had to put our own effort into the project and to figure out the things we thought they were important for success.
A whole development process was passed through. During the different stages a lot of problems occurred, which had to be solved. The group was very useful for this procedure, because usually somebody had a new idea to ship around these difficulties and if only one looks at his own things, he is sometimes blind on one eye for another way.
The construction, testing and improvement of the mouse was a very enjoyable time. We all learned a lot of how to behave in a small group, work together and especially to exchange minds.
Group work must not result in a way that only one member does all the work. Therefore the different tasks should be divided equally among the group. Unfortunately this is not always possible.
The mouse itself was a great success and reached the 2^{nd} place at the race, beaten only by 1 second. So it can be said that the overall design led into a good solution.

10. Technical Data

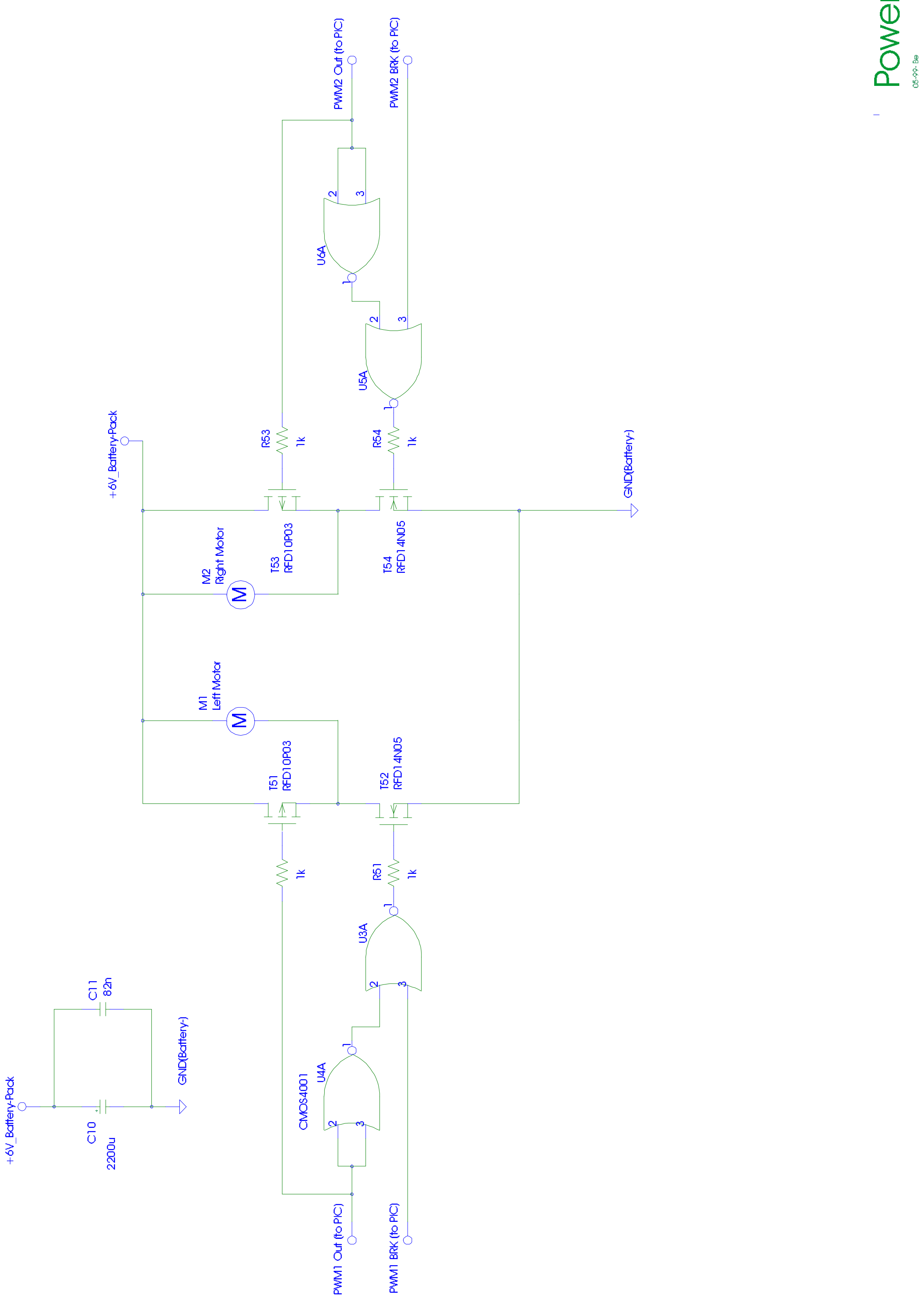
- 3V-6V operating voltage
- PIC 16F84 RISC Microcontroller
- Focused IR-Sensors (Optoelectronics OPB704W)
- 2x 16-bit PWM motor-control
- Braking capability
- MOSFET power stage
- Average speed 22cm/s on UEL standard maze (10s for 220cm)
- Compact size
- light weight

11. Appendix

a) Circuit diagram of the logic-board



b) Circuit diagram of the power-board



c) Used parts

Parts	Part Description	Price
1	**PIC16F84**	**3.74**
2	**LM339**	**1.60**
4	**RFD10P03**	**4.24**
4	**RFD14N05**	**2.32**
2	**Gearboxes incl. Motors**	**9.98**
2	**Wheels**	**3.58**
1	**Clad-board**	**0.40**
4	**Sensors OPB 704**	**11.64**
1	**X-tal 10Mhz**	**0.68**
2	**Tyres**	**1.98**
1	**Battery box**	**1.49**
4	**Batteries**	**2**
2	**Switch**	**1.02**
	Blutack	
	S U M	**44.67**

d) The complete Code

```
; Micromouse control and PWM Software
;
; Dirk Becker, 30/04/99, Race - version
; Sonja Lenz
; Michael Gims
;
; Group B2
;
; Port RA1, RA0 as input (Speed up/down)
; Port Rpwm2, RB4 as PWM output
;
;
; RA00 --> Sensor 3 (all Sensor inputs are active High)
; RA01 --> Sensor 4
; RA02 --> Start/test engine button (Active High)
; RA03 --> Sensor 5
; RA04 --> Pulse IR-diode out (via FET) (active Low output)
; RB00 --> Sensor 6
; RB01 --> Sensor 7
; RB02 --> PWM 1 - Brake Out (High= Brake ON)
; RB03 --> PWM 2 - Brake Out (High= Brake ON)
; RB04 --> PWM 1 - Output (High= Pulse ON) (Left )
; RB05 --> PWM 2 - Output (High= Pulse ON) (Right)
; RB06 --> Sensor 1
; RB07 --> Sensor 2
;
;
;     Sensor Location (Bit locations)
;
;     +-------------------------------------+
;     I                                     I
;     I    L                    4           I
;     I    M                    0           I
;     I    1                    1           I
;     I                         1           I
;     I                                     I     front
;     I    L                    P           I  ---------->
;     I    M                    I           I
;     I    2                    C           I
;     I                                     I
;     I                                     I
;     +-----+-+-----------------------+-+--+
;           I I                       I I
;           I I                       I I
;           I I                       S-2
;           S-1                       S-0
;           I I                       S-3
;           +-+                       +-+
;
; The PWM and Brake outputs are hardware protected against
; not allowed conditions in order to prevent the
; transistors from damage
;
;
;

      LIST    P=16f84;f=inhx8m


ERRORLEVEL       -302    ; suppress bank selection messages
_CP_OFF          equ   H'3FFF'                 ;code protect off
_PWRTE_ON   equ   H'3FFF'          ;Power on timer on
_WDT_OFF    equ   H'3FFB'                ;watch dog timer off
_HS_OSC          equ   H'3FFE'                 ;crystal oscillator
__CONFIG         _CP_OFF & _PWRTE_ON & _WDT_OFF &_HS_OSC
; *************************************
; *            DEFINITIONS            *
```

```
; *************************************

pcl   equ 2             ; Registers

status      equ 3
porta equ 5
portb equ 6
intcon      equ 0bh
trisa equ 85h
trisb equ 86h
optreg      equ 81h
tmr0  equ 01h

c     equ 0             ; Bits in status
z     equ 2
rp0   equ 5

out1  equ 4             ; PWM Output Bits
out2  equ 5
brk1  equ 2                   ; Brake Ouput Bits
brk2  equ 3

btn   equ 2             ; Button (Port A) Pressed=1
diode equ 4             ; Pulse Out for IR-Diodes


toif  equ 2             ; Bits in intcon

w     equ 0             ; Register destinations
f     equ 1             ;


stackw      equ 0ch                 ; stack to push pop the W-register ;
Variables
stacks      equ 0dh           ; stack to push pop the Status-register
pwm1  equ 0eh                 ; Ram address for PWM1 setting
pwm2  equ 0fh           ; Ram address for PWM2 setting
pwmhelp equ 010h        ; Ram address for PWM settings saving
n     equ 011h          ; Ram address for variable n
m     equ 012h          ; Ram address for variable m
k     equ 013h          ; ---       ""        ---  k
pulse equ 014f          ; Helping counter for pulses
sensors equ 015h        ; For IR-sensor
walls equ 016h          ; Count number of walls
l     equ 017h          ; Ram address for variable l

pwmmax      equ .230          ; PWM maximum speed
wait  equ .2                  ; Wait delay time
maxspd      equ .7                  ; Max speed the mouse can drive
wallcnt equ .20               ; Walls to drive around
goto init
```

```
; ***********************************************************
; *                 ISR Routine                          *
; * needs 0..F in pwm1 and pwm2 for generating pwm pulses  *
; * outputs directly to the motors                         *
; ***********************************************************

      org 04h
pwmisr
      movwf stackw            ; copy W-register to save stack
      swapf status, w   ; Swap Status-reg to be saved in W
      movwf stacks            ; Save Status to stacks


      movf  pwm1, w           ; Compare pwmhelp-counter
      subwf pwmhelp, w  ; with PWM1 setting
      btfsc status, c   ; and set PWM1 Output to
      bcf   portb, out1 ; on or
      btfss status, c
      bsf   portb, out1 ; off

      movf  pwm2, w           ; Compare pwmhelp-counter
      subwf pwmhelp, w  ; with PWM1 setting
      btfsc status, c   ; and set PWM1 Output to
      bcf   portb, out2 ; on or
      btfss status, c
      bsf   portb, out2 ; off

      incf pwmhelp, f         ; Increase PWM reference counter
      bcf pwmhelp, 4          ; But not >0Fh

      movlw pwmmax            ; sets w register
      bcf   status, rp0 ; select bank 0
      movwf tmr0        ; Set TMR0 to desired PWM resolution value



      bcf   intcon, toif      ; Clear interrupt flag
      swapf stacks, w   ; Swap nibbles in stacks reg
                        ; and place into W
      movwf status            ; restore Status-register
      swapf stackw, f   ; Swap nibbles in stackw and place into stackw
      swapf stackw, w   ; swap nibbles in stackw and restore to W-register
      retfie

      ; ISR END
```

```
; ***************************************************
; *   The initalisation is all done here        *
; ***************************************************

init
      bcf   status, rp0 ; select bank 0
      clrf  porta       ; set porta 0
      bsf   status, rp0 ; select bank 1
      movlw       0fh         ; set port A0 & A1
      movwf       trisa       ; as input

      bcf   status, rp0 ; select bank 0
      clrf  portb       ; set porta 0
      bsf   status, rp0 ; select bank 1
      movlw       0c3h        ; set port pwm1 & pwm2 & BRK1 & BRK2
      movwf       trisb       ; as output

      bsf   status, rp0 ; select bank 1
      movlw 40h         ; select TMR0
      movwf optreg            ; as counter with no divider

      bcf   status, rp0 ; select bank 0
      movlw 0e0h        ; Enable TMR0
      movwf intcon            ; interrupts

      clrf pwmhelp            ; Clear PWMHELP address,
      clrf pwm1         ; PWM1
      clrf pwm2         ; and PWM2

      clrf n                  ; clear variables
      clrf m
      clrf k
      clrf sensors

      bsf porta, diode  ; Turn off IR-diodes


start

      movlw 0           ; sets w register
      bcf   status, rp0 ; select bank 0
      movwf tmr0        ; Set TMR0 to desired PWM resolution value
```

```
; **********************************************************
; *        M A I N   P R O G R A M M E                     *
; **********************************************************

main

      bsf   porta, diode      ; Turn Off diodes (It's safer)

      movlw wallcnt           ; Set wallcounter
      movwf walls

      call  btnpress
      bcf   portb, brk1
      bcf   portb, brk2
      movlw 00h
      movwf pwm1
      movwf pwm2

; ***************************** The Mouse routines start here
***************

doagain

      call  straight          ; Go ahead
      call  turn              ; Drive the curve
      call  stop
      call  delay

      decfsz      walls, f
      goto  doagain

      goto main ; End main function ------------------------------

; **********************************************************
; *              Procedure Btnpress                      *
; * waits until Starting Button is pressed once and        *
; * released                                           *
; **********************************************************
;
btnpress

      btfss porta, btn  ; is start
      goto btnpress           ; button pressed?
      nop

      movlw 50          ; ca 50us
      movwf k                 ; loop for
loop2
      incfsz k, f       ; preventing
      goto loop2        ; button from jittering


loop1
      btfsc porta, btn  ; is the button
      goto loop1        ; released again?
      nop

      return
```

```
; *******************************************************
; *        Procedure Checksensors                  *
; * checks the IR-Diodes and writes the result into the    *
; * so called sensor variable sensor - sensor1 is bit 0        *
; * it also provides all the timing needed for the sensors *
; *******************************************************
;
checksensors


      clrf  sensors           ;
      bcf   porta, diode      ; Turns the diodes ON
      movlw       .20         ; Set counter register
      movwf       k                 ; to delay
loop4
      decfsz      k, f        ; for sensor
      goto  loop4       ; checking

      bsf   porta, diode      ; Turn the Sensors off again

      btfsc portb, 7    ; Check sensor 1
      bsf   sensors, 0  ;
      btfsc portb, 6    ; Check sensor 2
      bsf   sensors, 1  ;
      btfsc porta, 1    ; Check sensor 3
      bsf   sensors, 2  ;
      btfsc porta, 0    ; Check sensor 4
      bsf   sensors, 3  ;
      movf  sensors, w

      movlw       0ffh        ; Delay to turn the sensors
      movwf       k           ; long enough off
loop5
      decfsz      k, f        ;
      goto  loop5       ;

      return


; *******************************************************
; *        Subroutine Stop                          *
; *   Stops the engine WITH setting brakes               *
; *******************************************************
;
stop


      bsf   portb, brk1 ;Stop both motors and brake them
      bsf   portb, brk2
      clrw
      movwf pwm1
      movwf pwm2


      return
```

```
; *******************************************************
; *              Subroutine Delay                 *
; *                provides a delay                     *
; *******************************************************
;
delay


      movlw .255
      movwf n

stopagain1

      movlw .146
      movwf k

stopagain2

      movlw 1
      movwf m

stopagain3
      incfsz      m, f
      goto  stopagain3


      incfsz      k, f
      goto stopagain2

      incfsz      n, f
      goto stopagain1



      return

; *******************************************************
; *              Subroutine straight                  *
; *         Drives straight along the wall              *
; *******************************************************
;
straight

      movlw .10
      movwf n

straighton

      bcf   portb, brk1 ; release the brakes
      bcf   portb, brk2

      call  checksensors

      clrw


      btfsc sensors, 0
      call  straightrun

      clrw

      btfsc sensors, 3  ; Has right sensor wall contact?
      call  slow2       ; --> Yes: Jump slow 2

      clrw

      btfsc sensors, 2  ; Has left sensor wall contact?
      call  slow1       ; ; --> Yes: Jump slow 2

      bcf   sensors, 1  ; Sensor 1 should not be in use
```

```
movf  sensors, w


btfss status, z
goto  straighton

decfsz      n, f
goto straighton

call  stop
call  delay


btfsc sensors, 0  ; Was it a gap?
goto  straight          ; --> Do it again

btfsc sensors, 2  ; ---- "" -----
goto  straight

btfsc sensors, 3  ; ---- "" -----
goto  straight

call  stop
call  delay


return
```

```
; *******************************************************
; *              Subroutine slow1                  *
; *   Drives PWM1, with checking sensor 2--> result in w  *
; *******************************************************
;
slow1

      movlw maxspd
      movwf pwm2

      movlw maxspd-2    ; If only outer sensor has wall contact steer left
      btfsc sensors, 0
      movlw       maxspd-1    ; If mid and outer sensor have contact steer
slightly
      movwf pwm1

      return

; *******************************************************
; *              Subroutine slow2                  *
; * Drives PWM2, with checking sensor 3 --> result in w        *
; *******************************************************
;
slow2
      movlw maxspd
      movwf pwm1

      movlw maxspd-2    ; If only outer sensor has wall contact steer right
      btfsc sensors, 0  ;
      movlw       maxspd-1    ; If mid and outer sensor have contact steer
slightly
      movwf pwm2

      return
```

```
; ***********************************************************
; *               Subroutine Turn                   *
; *    This subrutine drives right around the corner      *
; ***********************************************************
;
turn

      call checksensors ; Check 'em again

      bsf   portb, brk2 ; Stop the
      clrw              ; right wheel
      movwf pwm2        ; (BRAKE IT!)

      bcf   portb, brk1 ; Turn with Left wheel
      movlw       .4
      movwf pwm1

      btfss sensors, 3
      goto  turn


      return




      END               ; it's enough now
```