Course No: **CSE 408 N**

Course Title: **Computer Interfacing Sessional**

Taxi Meter

Design & Implementation of a Microcontroller based Taximeter

Group No: **2**

Group Members:	**Tamal Saha,**	**#0205002**
	Md. Jawaherul Alam,	**#0205003**
	Md. Mainul Mizan,	**#0205004**
	Hasan Shahid Ferdous,	**#0205014**
	A.K.M. Golam Sarwar,	**#0205026**

Project Supervisor: **Md. Mostafa Ali Patwary**

Table of Contents

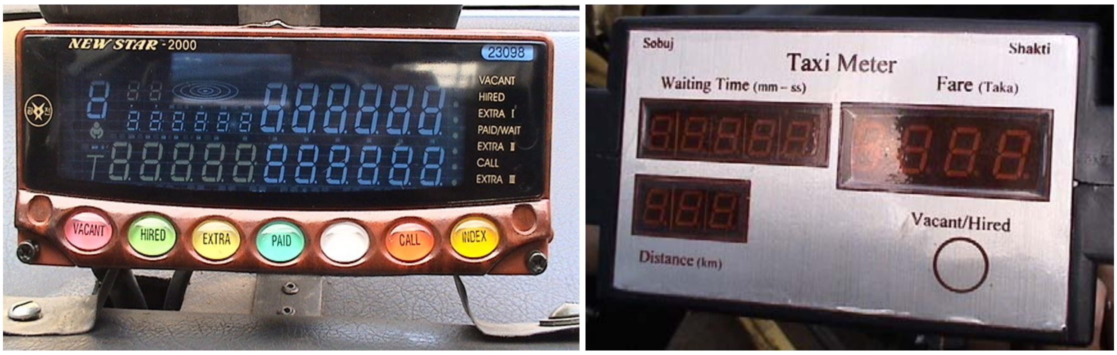
Introduction:

In our CSE 408N Course (Computer Interfacing Sessional) we took Taxi meter as the term project of our group (Group 2). Taxi meter is a device attached to the taxi that provides information during a trip. We all are familiar with taxi meters. CNG taxies, non air conditional and air conditional taxies are available in our country. Taxi meter is a part and parcel of these taxies. Currently, all the taxi meters in use are imported from abroad, typically from Korea and India. But, we can make it in our country, and by doing so, can save lots of foreign currency.

Problem description:

Our objective was to design and implement a microcontroller based taxi meter that performs all the operations available in the typical taxi meters available in our country and in use. Current taxi meters show information about distance traveled, waiting time and fare. There are two buttons to provide three functionalities:

1. Start: Starts taxi meter.
2. Pause: Pauses the meter
3. Turn off: Turns off the meter.



a. A taxi meter from Cab Salida A/C taxi cab

b. A taxi meter from CNG taxi

Figure: Typical taxi meters

The operation of a taxi meter is simple enough. When a customer hires a taxi, the driver starts the meter. The meter starts with a distance traveled set to zero, zero waiting time and a fixed starting fare which depends on the type of the taxi. In CNG's it is 12 tk, 15 tk. for non A/C taxi cabs and 20 tk. for A/C taxi meters. That means the minimum fare if one hires a taxi cab. There are fixed waiting time charge, fare per kilometer decided by the government. We followed the specifications of the CNG taxi cabs, that is initial fare is 12 tk, 6 tk. fare per km. for first 2 kilometers and 5 tk. per km. then, waiting charge is 0.50 tk per minute. If the driver needs to stop somewhere in the road (like taking fuel and so on), he may pause the meter. And after completion of the journey, he stops the meter. This is how taxi meter operates.

There are some other issues related to the operation of the taxi meter. Distance is calculated from the no of rotations of the wheel. As the radius of the wheel is fixed, its perimeter is the distance traveled in one rotation of the wheel. So we can calculate the distance traveled from the no of rotations of the wheel.

When a taxi stops for some reason (like traffic jam) or moves at a very slow speed (we set it for approximately 5km/hour and called it threshold velocity) then waiting time is calculated.

Fare is calculated by multiplying the traveled distance by fare rate if the velocity of the taxi is above that threshold velocity. If the velocity of the taxi meter is below that predefined threshold, then fare is calculated at a different rate based on the elapsed interval.

Block diagram with discussion:

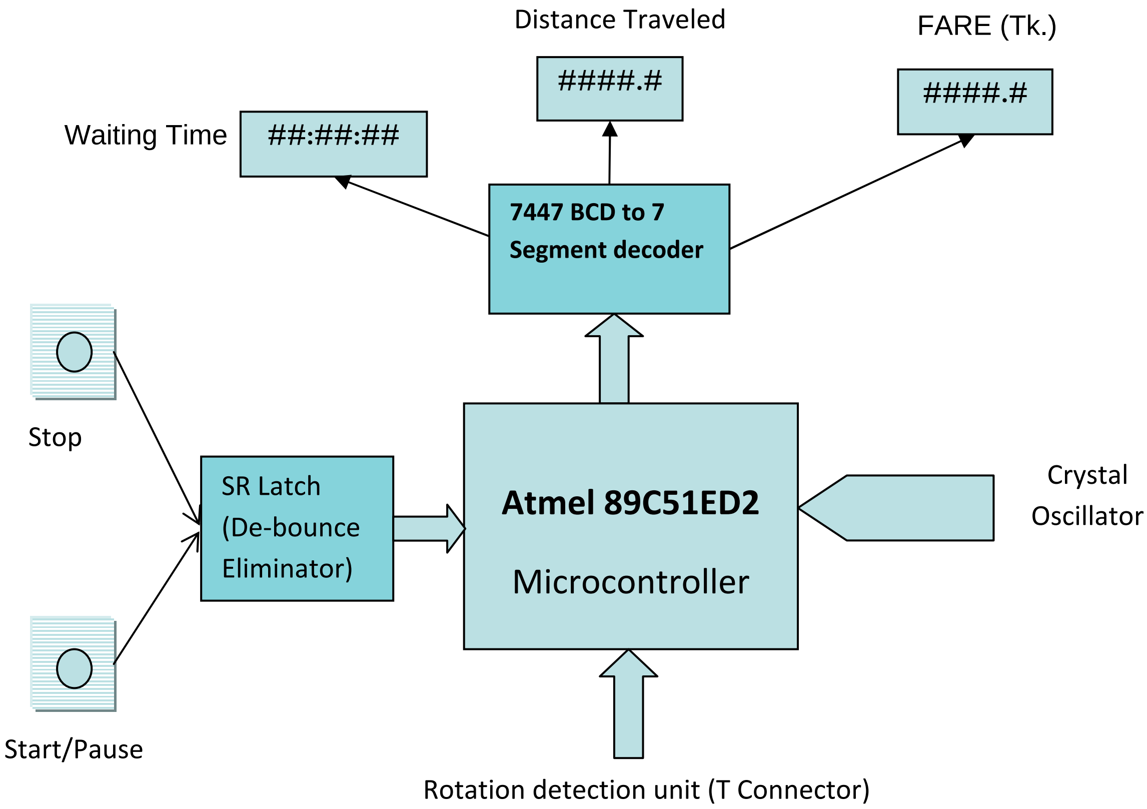


Figure: Block Diagram of Taxi Meter

There are some things that need to be explained regarding this block diagram. In a regular taxi cab, there is a line that comes from the gear box to a device named "T Connector". There is a metal head in front of that line that moves as much as the wheel. It goes to one end of that T Connector; another end of the T Connector goes to the mile meter and the T connector produces square wave which frequency equals the rotation of the wheel. Inside the T connector, there is a Hall Effect sensor that is used to detect rotation.

Figure: Metal head in front of the line coming from gear box

Figure: T Connector

Figure: One end of T Connector

Electrical components:

1. Atmel 89C51ED2 microcontroller
2. 74279 - SR Latch
3. through a 74154 - 4 to 16 line decoder
4. 7447 - BCD to seven segment decoder
5. Seven segment display - anode type
6. BD 136 - PNP type transistor
7. Capacitor: 30pf, 2µF
8. Resistor: 1kΩ, 68Ω, 20Ω
9. Crystal Oscillator: 12 MHz
10. Push button

Pin diagram with discussion:

We can describe different parts of the pin diagram.

One BCD to seven segment decoder is used to control all the 16 seven segment displays. So this is a software controlled seven segment display configuration. A 4 to 16 line decoder is needed to enable seven segment displays one by one. The transistors BD-136 is used to switch on the seven segment displays when needed, 68Ω resistors with 7447 is required to limit the current segment when it is on.

The microcontroller we used was Atmel 89C51ED2. It is compatible with the Intel 8051. In addition to all the normal features of 8051 it has 64 Kilo Bytes of flash memory. It has 2 external interrupts and 3 timers. We used

Interrupt 0 for stop button, interrupt 1 for start button. Timer 0 is used for rotation count, timer 1 for slowdown event. We calculated that at the speed of 5 km/hour, the wheel of a taxi has 1 rotation is 0.6 second. We updated our display after 10 rotations. So we start timer 0 and timer 1 simultaneously. Timer 0 overflows after 10 rotations. Timer 1 after 6.785 milliseconds. When timer 1 overflows for 104 times, we decided that the taxi is moving at a speed below the threshold. If timer 0 overflows before it, taxi is moving at a speed above the threshold and timer 1 is reset.

The fare, distance and waiting time is calculated accordingly.

The microcontroller has three 8 bit ports. We used the lower 4 bits of port 0 to control which 7 segment is to light up and upper 4 bits of the port 0 to send display value to 7447.

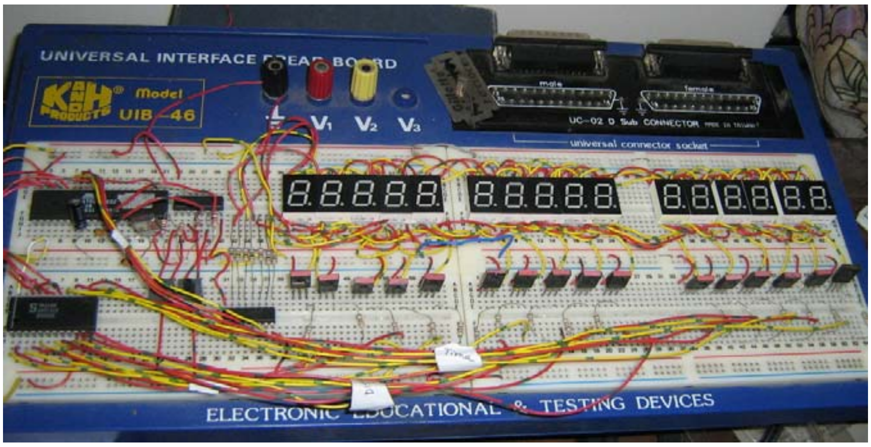

UNIVERSAL INTERFACE BREAD BOARD
K&H PRODUCTS
Model UIB-46
V1
V2
V3
UC-02 D Sub CONNECTOR
universal connector socket
ELECTRONIC EDUCATIONAL & TESTING DEVICES

Figure: Our taxi meter in bread board

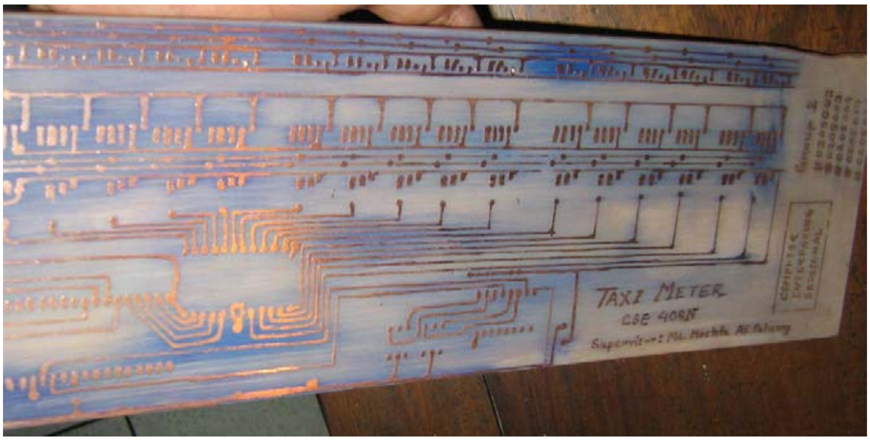


Figure: Our PCB layout design

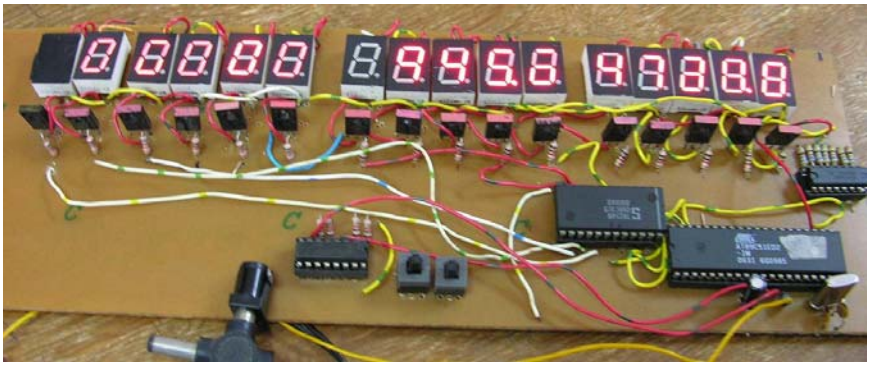


Figure: Our taximeter - after completion.

At first we implemented our project in bread board. Then we implemented it in PCB (Printed circuit board).We designed to PCB layout, draw it and we have done the drilling and soldering all by ourselves.

Problems faced and solution:

The first problem we faced was to choose the right microcontroller. The microcontroller programmer in our lab could not program the microcontroller we used. Fortunately, we got a microcontroller programmer from one of our junior students in the hall.

We got a drill machine in our lab and made our PCB board. For chemical compounds required to make PCB, we depend on the local market.

Future Extension:

LCD displays may be used instead of 7 – segment displays. The same microcontroller can be used to interface with the LCD display. We used a flash memory based microcontroller. But in the real world we should use a ROM based microcontroller so that its contents cannot be altered. We did so because ROM based microcontroller writer was not available in our Lab.

Conclusion:

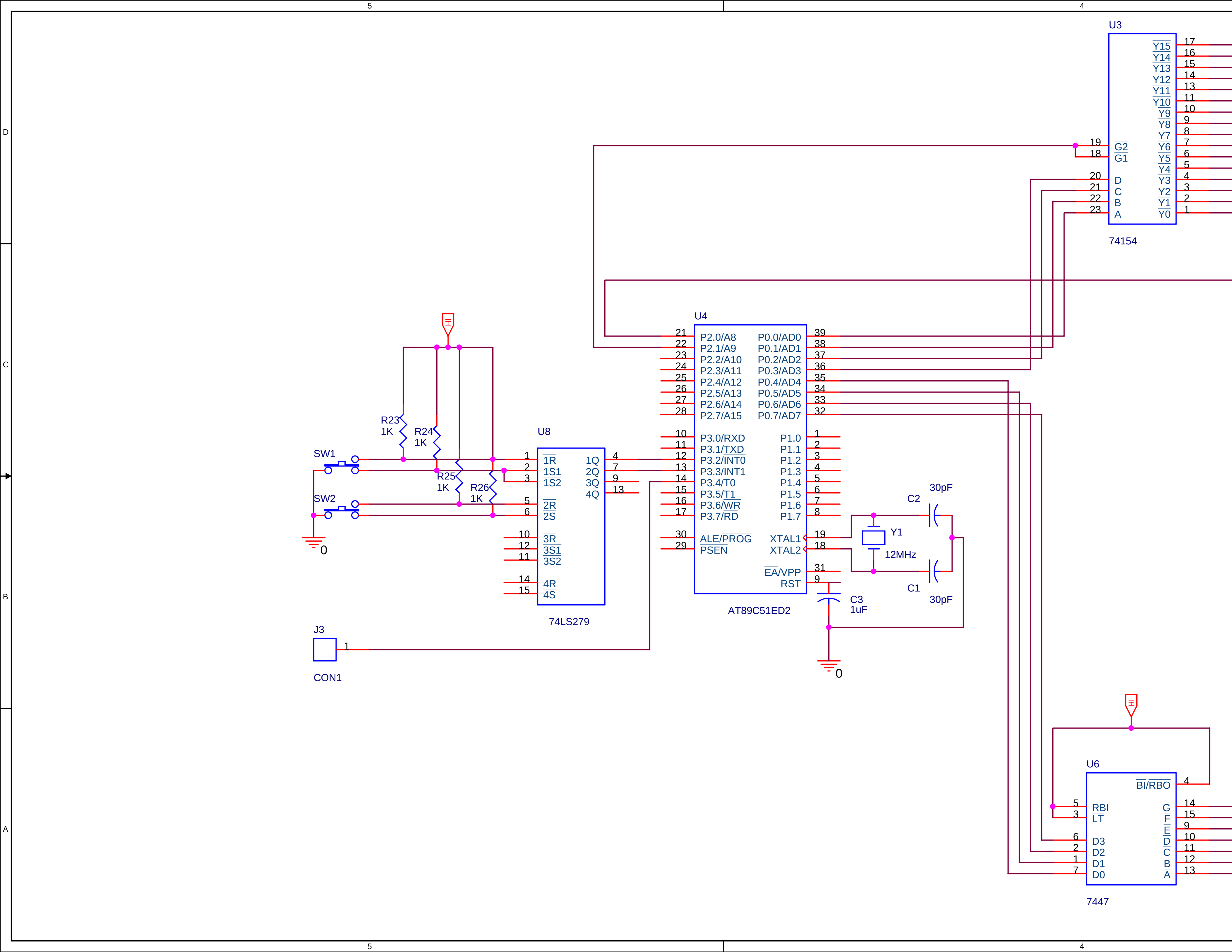
This was a successful project and we believe we have overcome all the difficulties and reached the functionalities of typical taxi meters that are in use in our country. All the parts needed are available in our country and collected from our local market. After a successful field test, our Government can take steps to implement this in our taxi meters and we can save a lot of foreign currency. Now, we import these taxi meters at a cost of 600 tk to 12000 tk. But we can make it here at a cost of 700-1000 tk only.

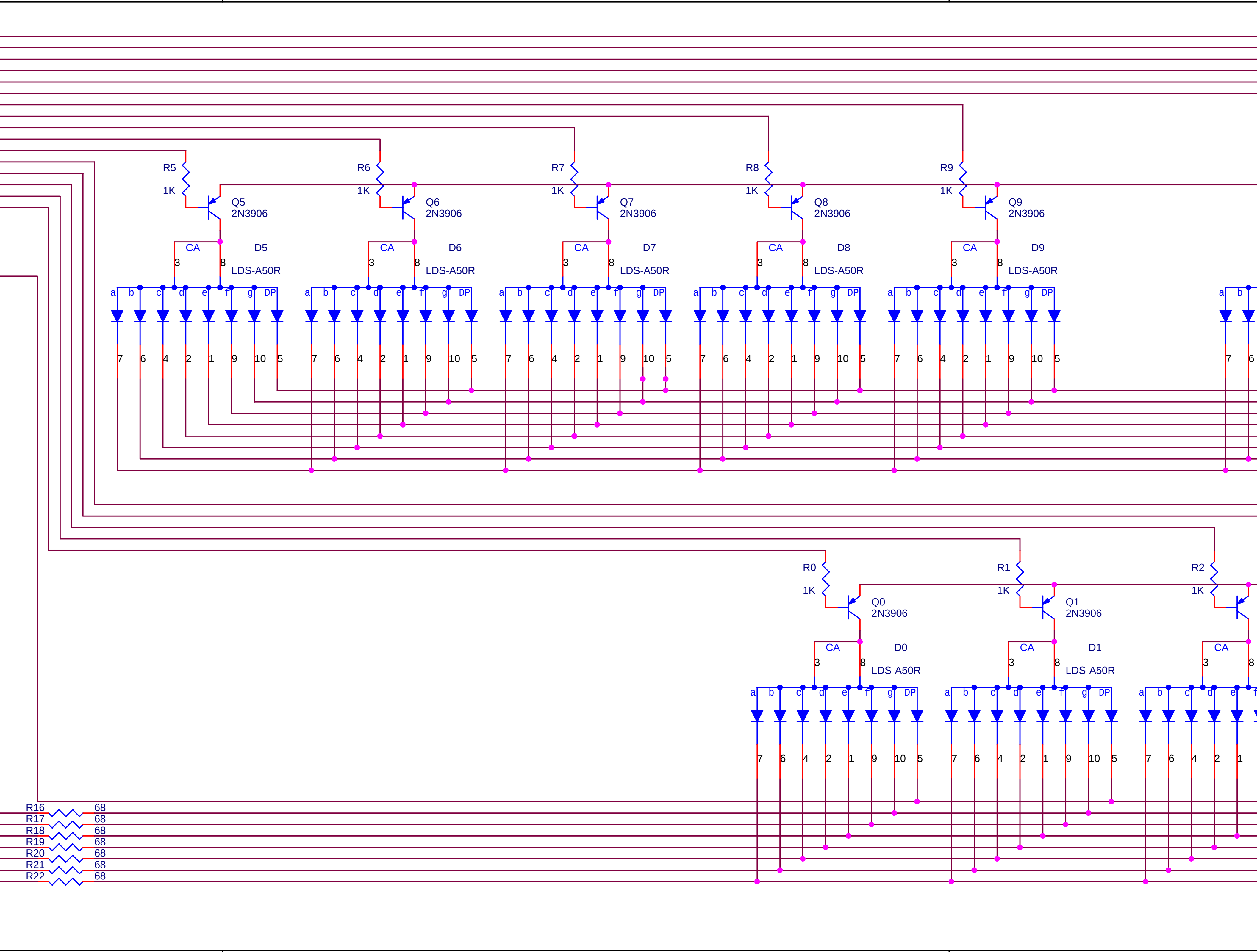
References:

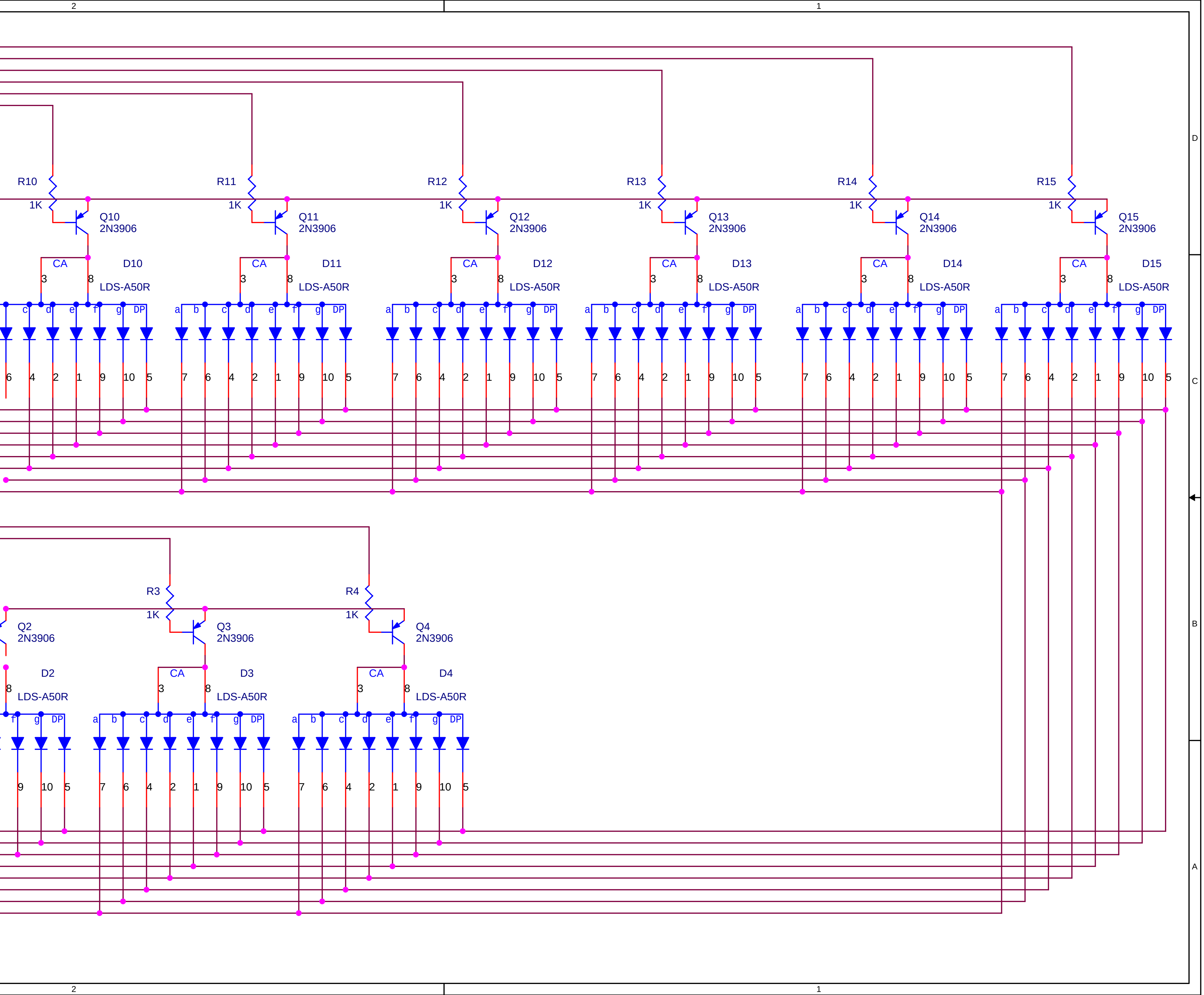
1. www.minimopar.net
2. Microprocessor data handbook
3. www.atmel.com

Appendix A: Pin Diagram of Taximeter

The pin diagram of the taximeter is included here.


5
4
D
C
B
A
U3
74154
Y15
Y14
Y13
Y12
Y11
Y10
Y9
Y8
Y7
Y6
Y5
Y4
Y3
Y2
Y1
Y0
G2
G1
D
C
B
A
U4
AT89C51ED2
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
ALE/PROG
PSEN
XTAL1
XTAL2
EA/VPP
RST
U8
74LS279
1R
1S1
1S2
2R
2S
3R
3S1
3S2
4R
4S
1Q
2Q
3Q
4Q
SW1
SW2
R23 1K
R24 1K
R25 1K
R26 1K
C1 30pF
C2 30pF
C3 1uF
Y1 12MHz
J3
CON1
0
0
U6
7447
BI/RBO
RBI
LT
D3
D2
D1
D0
G
F
E
D
C
B
A


R5
1K
Q5
2N3906
CA
D5
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
R6
1K
Q6
2N3906
CA
D6
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
R7
1K
Q7
2N3906
CA
D7
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
R8
1K
Q8
2N3906
CA
D8
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
R9
1K
Q9
2N3906
CA
D9
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
a b
7 6
R0
1K
Q0
2N3906
CA
D0
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
R1
1K
Q1
2N3906
CA
D1
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
R2
1K
CA
3
8
a b c d e f
7 6 4 2 1
R16 68
R17 68
R18 68
R19 68
R20 68
R21 68
R22 68


2
1
D
C
B
A
R10
1K
Q10
2N3906
CA
D10
3
8
LDS-A50R
c d e f g DP
6 4 2 1 9 10 5
R11
1K
Q11
2N3906
CA
D11
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
R12
1K
Q12
2N3906
CA
D12
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
R13
1K
Q13
2N3906
CA
D13
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
R14
1K
Q14
2N3906
CA
D14
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
R15
1K
Q15
2N3906
CA
D15
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
Q2
2N3906
D2
8
LDS-A50R
f g DP
9 10 5
R3
1K
Q3
2N3906
CA
D3
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5
R4
1K
Q4
2N3906
CA
D4
3
8
LDS-A50R
a b c d e f g DP
7 6 4 2 1 9 10 5

Appendix B: Taximeter Code

```
/**********************************************************************************************
*                          T  A  X  I        M  E  T  E  R
*
*                                 MCU Family: 8051                                           *
*                                 MCU:  AT89C51ED2
*
 **********************************************************************************************/

/*
 * Group No.     : 2
 * Group Members: 0205002,
 *                0205003,
 *                0205004,
 *                0205014,
 *                0205026
 * Author        : Tamal Saha, #0205002, Email: saha_tamal2002@yahoo.com, tamal.saha@gmail.com
 * Date          : 20-01-2007
 */

/*
 * Assumptions:
 *       1.      External interrupt 0 used for STOP Button
 *       2.      External interrupt 1 used for START/PAUSE Button
 *       3.      Timer 0 used for rotation count
 *       4.      Timer 1 used for slowdown timer
 *       5.      Diameter of wheel of Taximeter = 30 cm = 0.3 m
 *       6.      pi = 3.141592654
 *       7.a.    Max velocity of Taximeter = 80 km/h
 *       7.b.    Max rps = 24
 *       8.      Rotation count incremented every 10 rotation
 *       9.a.    Slowdown timer threshold velocity = 5 km/h
 *       9.b.    Waiting time incremented if slowdown timer value >= 6.785 sec
 */
/*---------------------- *
 * I N C L U D E S       *
 *---------------------- */

#include "reg_c51.h"

/*---------------------- *
 * D E F I N E S         *
 *---------------------- */

#define MIN_FARE                 120             // 12.0 Tk. minimum fare
#define DISTANCE_PER_ROTATION    0.942477796     // 0.942477796 m/rotation

#define DISTANCE_LEVEL_1    20          // 2.0 km
#define FARE_LEVEL_1        6          // FARE rate 6.0 Tk/km for first 2.0 km
#define MAX_LEVEL_1_FARE    120        // 12.0 Tk. Eqn: DISTANCE_LEVEL_1 * FARE_LEVEL_1 /100
#define FARE_LEVEL_2        5          // FARE rate 5.0 Tk/km after first 2.0 km
#define WAITING_FARE_RATE   2          // 2.0 Tk./min of waiting time


#define ROTATION_PER_OVERFLOW    10
#define TH0_AUTO_RELOAD_VALUE    246   //Timer 0 auto reload value = 256 - ROTATION_PER_OVERFLOW

#define MIN_SLOWDOWN_TIME        6785 // in ms; Maximum time for 10 rotation

#define FULL_OVERFLOW_COUNT      103  // 6.785840132/65536E-06 = 103.5437032

#define TH1_DEFAULT_INIT_VALUE   0    // TH1 = 0;   Timer 1 normally starts from 0
#define TL1_DEFAULT_INIT_VALUE   0    // TL1 = 0;
```

```
#define TH1_LAST_INIT_VALUE      116   // TH1 = 116; Timer 1 starts from 29904(= 116*256 + 208) during
                                       // FULL_OVERFLOW_COUNT overflows
#define TL1_LAST_INIT_VALUE      208   // TL1 = 208;
#define LAST_INIT_VALUE          29904 // 29904 = 116*256 + 208 = 65536- (6.785840132E+06 - 103*65536)

//Meter status values
#define ACTIVE  1
#define PAUSED  2
#define STOPPED 0

//DELAY(N); function
#define DELAY(N) for(i=0; i<N; i++)

/*-----------------------------------------*
 * G L O B A L  V A R I A B L E S          *
 *-----------------------------------------*/

unsigned char meterStatus;             // meterStatus in first 128 byte
unsigned char timer1OverflowCount;     // counts # of times Timer 1 overflows
bit hasSlowed;                         // vehicle has slowed down
unsigned int fractionSlowdownTime;     // in ms the fraction slowdown time in timer 1 counter

unsigned long rotationCount;           // # of rotations
unsigned long waitingTime;             // waiting time in ms

unsigned long distance;                // distance     in meter
unsigned long waitingHHMMSS;           // waiting time in ms
unsigned long fare;                    // fare         in paisa

unsigned char P_Dummy;                 // dummy port data
char index;                            // 7 segment display index : FARE:0-4, DISTANCE:5-9,
                                       // WAITING_TIME: 10-11: 12-13: 14-15
```

```
unsigned char digitShowed;            // flag used to show at least 2 digits
unsigned char temp;                   // temporary variable used for showing time
unsigned char i;                      // delay loop counter

void main(void)
{
    //interrupt configuration

    IT0 = 1;      //low level triggered external interrupt 0
    IT1 = 1;      //low level triggered external interrupt 1
    IPL0 = 0x01;  //interrupt priority = 0 0 0 0 0 0 0 1
    IEN0 = 0x8F;  //interrupt enable   = 1 0 0 0 1 1 1 1

    //interrupt configuration ends

    while(1)
    {
        if(meterStatus != STOPPED)
        {
            distance = (rotationCount*DISTANCE_PER_ROTATION + 50)/100;  // in meter
                                                        // 50 added for ceiling
                                                        // distance approximated to one digit meter
            waitingHHMMSS = (waitingTime + 500)/1000;   // in ms;
                                                        // 500 added for ceiling
                                                        // waiting time approximated to one digit second

            fare = waitingHHMMSS * WAITING_FARE_RATE / 6;     // fare for waiting time in one digit paisa

            if(distance <= DISTANCE_LEVEL_1)
            {
                fare += distance * FARE_LEVEL_1;            // fare
            }
            else
```

```
{
     fare += (MAX_LEVEL_1_FARE + (distance - DISTANCE_LEVEL_1)* FARE_LEVEL_2);
}

if(fare < MIN_FARE)
{
     fare = MIN_FARE;
}

// display F A R E
index = 0;                              // 7 segment display index set to LSD of FARE
digitShowed = 0;
while( (digitShowed < 2 || fare > 0) && index <= 4) //show at least two digits OR until fare > 0
{
     //index >0 is just for overflow(FARE >= 9999.9 Tk) protection
     P_Dummy = fare%10;
     P_Dummy = P_Dummy << 4;
     P_Dummy |= index;
     P0 = P_Dummy;               // out fare digit
     P2_0 = index == 1 ? 0 : 1;  // set decimal point for 7 Seg #3

     fare /= 10;
     index++;
     digitShowed++;
     DELAY(25);
}
// display F A R E ends

// display D I S T A N C E
index = 5;         // 7 segment display index set to LSD of DISTANCE
digitShowed = 0;
while((digitShowed < 2 || distance > 0) && index <= 9)
{
      P_Dummy = distance%10;
      P_Dummy = P_Dummy << 4;
      P_Dummy |= index;
      P0 = P_Dummy;                    // out distance digit
      P2_0 = index == 6 ? 0 : 1;       // set decimal point of 7 Seg #8

      distance /= 10;
      index++;
      digitShowed++;
      DELAY(25);
}
// display D I S T A N C E ends

// display W A I T I N G  T I M E

//display S E C O N D
temp = waitingHHMMSS%60;   // temp set to second
index = 10;                // 7 segment display index set to LSD of SECOND
while(index <= 11)
{
      P_Dummy = temp%10;
      P_Dummy <<= 4;
      P_Dummy |= index;
      P0 = P_Dummy;          // out 'SECOND' digit
      P2_0 = 1;              // no decimal point

      temp /= 10;
      index++;
      DELAY(25);
}

//display M I N U T E
waitingHHMMSS /= 60;
```

```
                temp = waitingHHMMSS%60;    // temp set to minute
                while(index <= 13)
                {
                      P_Dummy = temp%10;
                      P_Dummy <<= 4;
                      P_Dummy |= index;
                      P0 = P_Dummy;           // out 'MINUTE' digit
                      P2_0 = 1;               // no decimal point

                      temp /= 10;
                      index++;
                      DELAY(25);
                }

                //display H O U R
                waitingHHMMSS /= 60;
                while(index <= 15)
                {
                      P_Dummy = temp%10;
                      P_Dummy <<= 4;
                      P_Dummy |= index;
                      P0 = P_Dummy;           // out 'HOUR' digit
                      P2_0 = 1;               // no decimal point

                      temp /= 10;
                      index++;
                      DELAY(25);
                }
                // display W A I T I N G  T I M E ends
          }
    }
}
```

```
/*
 *    Purpose: External interrupt 0 is used to stop the meter
 *
 */
void external0(void) interrupt 0 using 0
{
      P3_2 = 1;   //remove the low level triggered signal from P3.2 pin

      TR0 = 0;   //stop timer 0(rotation counter)
      TR1 = 0;   //stop timer 1(slowdown counter)
      P2_1 = 1;   //74154 Decoder is disabled to stop power loss through the 7 Segment Display

      meterStatus = STOPPED; //meter status set to stopped
}


/*
 *    Purpose: External interrupt 1 is used to handle Start/Pause tole button
 *
 */
void external1(void) interrupt 2 using 0
{
      P3_3 = 1;       //remove the low level triggered signal from P3.3 pin

      if(meterStatus == ACTIVE)
      {
            TR0 = 0;   //stop timer 0(rotation counter)
            TR1 = 0;   //stop timer 1(slowdown counter)

            meterStatus = PAUSED; //meter status set to pause
      }
      else if(meterStatus == PAUSED)
      {
            meterStatus = ACTIVE; //meter status set to active

            TR0 = 1; //start timer 0(rotation counter)
            TR1 = 1; //start timer 1(slowdown counter)
        }
        else            //if(meterStatus == STOPPED)
        {
            //initialize rotation & waiting time counter
            rotationCount = 0;
            waitingTime = 0;

            //timer configuration

            TMOD = 0x16; //timer0 GATE = 0, counter, mode 2 (auto reload)
                                //timer1 GATE = 0, timer,   mode 1 ()
            TH0 = TH0_AUTO_RELOAD_VALUE;  //interrupt in every 10 rotations
            TL0 = TH0_AUTO_RELOAD_VALUE;

            TL1 = TL1_DEFAULT_INIT_VALUE;    // clear low timer1
            TH1 = TH1_DEFAULT_INIT_VALUE;    // clear high timer1
            timer1OverflowCount = 0;               // clear timer1 overflowCount
            hasSlowed = 0;                   // cleared at start of timer

            //timer configuration ends

            P2_1 = 0;   //74154 Decoder is enabled to give power to the 7 Segment Display
            meterStatus = ACTIVE; //meter status set to active

            TR0 = 1; //start timer 0(rotation counter)
            TR1 = 1; //start timer 1(slowdown counter)
        }
    }
```

```
/*
 *   Purpose: Since Timer 0 is configured to work as counter, an overflow occurs every
 *   10 rotation. We stop the slowdown timer(timer 1). Then timer 1 is configured
 *   to start count from 0. Then timer 1 is restarted. */

void timer0(void) interrupt 1 using 0
{
     TR1 = 0;   // stop timer1 (slowdown timer)

     if(hasSlowed == 1)
     {
          if(timer1OverflowCount > 0)
          {
               waitingTime += (((timer1OverflowCount - 1) * 65536)/1000);
          }

          fractionSlowdownTime = TH1;
          fractionSlowdownTime <<= 4;
          fractionSlowdownTime |= TL1;

          if(timer1OverflowCount > FULL_OVERFLOW_COUNT)
          {
               fractionSlowdownTime -= LAST_INIT_VALUE;
          }

          waitingTime += (fractionSlowdownTime/1000);
          hasSlowed = 0;
     }

     TL1 = TL1_DEFAULT_INIT_VALUE;    // clear low timer1
     TH1 = TH1_DEFAULT_INIT_VALUE;    // clear high timer1
     timer1OverflowCount = 0;              // clear timer1 overflowCount
```

```
      rotationCount += ROTATION_PER_OVERFLOW; // ten more rotations completed

      TR1 = 1;   // start timer1 (slowdown timer)
}
/*
 *   Purpose: Since Timer 0 is configured to work as counter, an overflow occurs every
 *   10 rotation. We stop the slowdown timer(timer 1). Then timer 1 is configured
 *   to start count from 0. Then timer 1 is restarted.
 *
 * Calculation:
 *
 *             3600 * pi * diameter_wheel          [diameter_wheel in meter]
 * 1 rotation = -------------------------  sec
 *                  max_velocity*1000               [max_velocity in km/h]
 *
 *           = 0.6785840132 sec where diameter_wheel = 0.3m, max_velocity = 5 km/h
 *
 * 10 rotation = 6.785840132 sec
 *             = 103 full range overflow + 1 35632(29,904-65,535) range overflow
 *
 */
void timer1(void) interrupt 3 using 0
{
      timer1OverflowCount++;

      if(timer1OverflowCount == FULL_OVERFLOW_COUNT)
      {
            TH1 = TH1_LAST_INIT_VALUE;
            TL1 = TL1_LAST_INIT_VALUE;
      }
      else if(timer1OverflowCount > FULL_OVERFLOW_COUNT)
      {
            waitingTime += MIN_SLOWDOWN_TIME;
            timer1OverflowCount = 0;
            hasSlowed = 1;
      }
}
```