پایان نامه کارشناسی

در رشته الکترونیک

موضوع:

برد ارتباطی USB ، Wifi ، SMS و BlueTooth

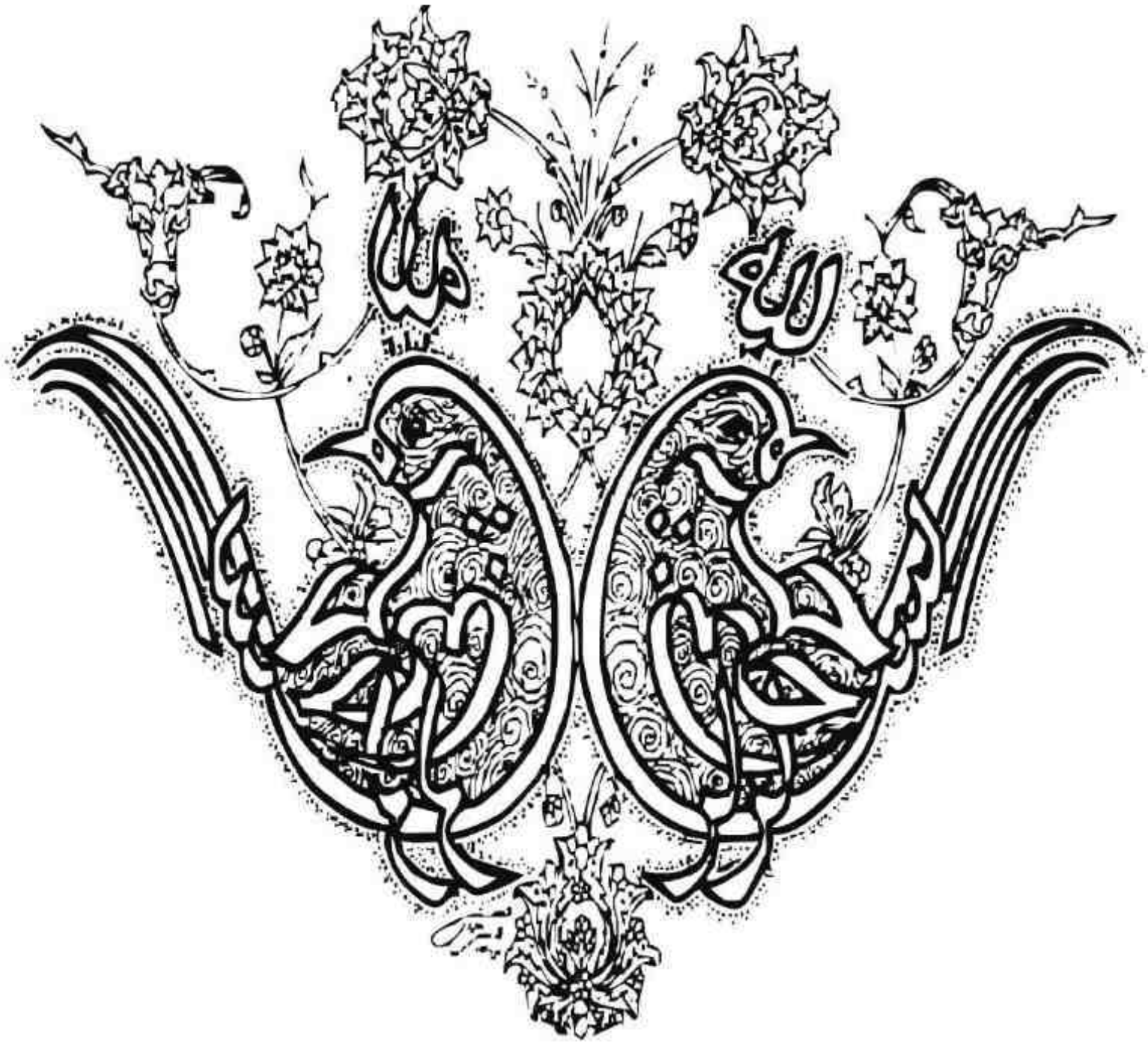
استاد راهنما

دکتر سید حسین پیشگر

نگارش:

آرش فتاحی

زمستان ۱۳۹۴



پایان نامه کارشناسی

در رشته الکترونیک

موضوع:

برد ارتباطی USB ، Wifi ، SMS و BlueTooth

استاد راهنما

دکتر سید حسین پیشگر

نگارش:

آرش فتاحی

زمستان ۱۳۹۴

تقدیم به

وب سایت www.Prozhe.com

فهرست مطالب

فهرست مطالب

فهرست جداول

فهرست اشکال

فهرست اشکال

عنوان	صفحه

چکیده

امروزه ترمینال های بیسیم جهت برقراری ارتباط بین دو دستگاه که هر دو از سیستم مدیریتی ارتباط بیسیم استفاده می کنند بسیار رواج پیدا کرده است. دو دستگاه دارای ارتباط بیسیم پس از پیدا کردن یکدیگر می توانند با پروتکل های گوناگون شروع به داد و ستد اطلاعات کنند.سامانه های تعریف شده امروزه تاکید زیادی برای ایجاد ارتباطات بیسیم برای کاهش هزینه ها و سرعت عمل بیشتر در مخابره کردن اطلاعات دارد.ترمینال های استفاده شده می توانند به استفاده از پروتکل سریال شروع به مبادله اطلاعات از هر نقطه و فاصله معینی کنند.

واژگان کلیدی: ارتباطات بیسیم ،میکرو کنترلر ATmega128، ماژول های ارتباطات سریال

فصل اول

کلیات پروژه

۱-۱- مقدمه

تحول تکنولوژی در زمینه های ارتباط بیسیم باعث گردیده تا روز به روز اشتیاق به کم کردن ارتباطات سیمی و افزایش و طراحی دستگاه هایی که حجم زیادی از اطلاعات را به صورت بیسیم مخابره می کنند گردد. از این نوع ارتباطات می توان به ارتباطات بیسیم که توسط گوشی های موبایل صورت می گیرد اشاره کرد. ارتباطاتی همچون ارسال پیامک می تواند تا کیلومتر ها آن طرف تر رد و بدل گردد و کنترل بسیاری از دستگاه های ساده را از فاصله دور ممکن کند. با قرار دادن سیستم های BMS در منزل، به طور مثال می توان از طریق ارسال و دریافت پیامک از وضعیت های مختلف خانه همچون دما، دزدگیر ها و ... مطلع گردید. همچنین می توان از این طریق پیش از رسیدن به خانه وضعیت مطلوب را از طریق پیامک به روتر[1] BMS منزل پیامک کرده تا قبل از رسیدن به منزل وضعیت مطلوب از نظر دمایی، روشنایی و ... حاصل گردد. با پیشرفت سریع در مقوله ارتباطات و مخصوصا گوشی های موبایل امکان استفاده از روش های دیگر همچون بلوتوث و SMS میسر گردیده است که می توان از طریق یک اپلیکشین ساده به کنترل وسایل برقی با تنظیمات بسیار پیشرفته تر پرداخت. در کشور های توسعه یافته استفاده از این نوع اپلیکیشن ها و ماژول های ارتباطی روز به روز در حال توسعه می باشد که از مهم ترین دلایل می توان به ارزان بودن و کمتر کردن دستگاه های جانبی جهت کنترل این ماژول ها اشاره کرد.از اتصال USB نیز جهت کنترل قسمت های برد با رایانه می باشد.

[1]Building management system

۱-۲- بیان مساله پروژه:

در این پروژه بردی ساخته شده که قابلیت اتصال ماژول ها مربوط به وای فای ، بلوتوث ، پیامک و USB [1]را دارد. هدف این است که با اتصال این برد به رایانه های شخصی امکان کنترل قسمت های مختلف برد به صورت سیمی امکان پذیر گردد و همچنین می توان از طریق یک گوشی مجهز به سیستم عامل آندروید اقدام به کنترل قسمت های مختلف برد اعم از رله ها، LCD ها، صدای بوق ، LED های لحیم شده روی برد و همچنین اتصال قطعات جانبی دیگر به آن استفاده کرد.

۱-۳ اهداف پروژه:

هدف اصلی در این پروژه ایجاد ارتباط بین برد ساخته شده با رایانه و گوشی موبایل می باشد که به صورت زیر بیان می شود:

۱- ایجاد ارتباط با SMS [2] و تغییر در وضعیت قطعات موجود بر روی برد.

۲- استفاده از برنامه ترمینال بلوتوث جهت ارتباط با برد و ارسال داده به میکرو.

۳- استفاده از برنامه ترمینال وای فای برای ارتباط با برد.

٤- اتصال برد به رایانه از طریق پروتکل ارتباطی USB و مبدل سریال به USB

[1] Universal Serial Bus
[2] Short Message Service

فصل دوم

قطعات و ماژول های استفاده شده در برد

۲-۱- شرح کلی قطعات

در این پروژه از میکروکنترلر شرکت ATMEL با پارت نامبر Atmega128 استفاده گردیده است که برای این پروژه به دلیل نیاز به پایه های زیاد جهت کنترل قسمت های مختلف برد بسیار کاربردی می باشد.یک LCD با کنترلر HD4470 مدل LM016L استفاده گردیده تا اطلاعات ارسالی و دریافتی از برد بر روی آن نوشته شود. این برد قابلیت اتصال LCD گرافیکی با تعداد پیکسل ۱۲۸ در ٦٤ نیز دارا می باشد. یک Buzzer[1] برای تولید صدا ، دو رله ٥ ولتی برای اتصال وسایل ولتاژ بالا برای کنترل ، چهار LED ، ماژول SIM900 برای اس ام اس ، ماژول HC-05 برای برقراری اتصال بلوتوث ، ماژوول ESP8266 جهت برقراری ارتباط وای فای ، آیسی FT232 به عنوان مبدل USB به SERIAL ٤ دکمه متصل به میکرو ، ٤ عدد کانکتور فونیکس برای اتصال وسایل برقی ولتاژ بالا، مدار تغذیه با خروجی سه ولتاژ ٥ ولت ، ۳,۳ ولت و ٤,۱ ولت و جای ارتباط NRF (رادیویی) بر روی برد موجود است. حال به بررسی تک تک قطعات موجود بر روی برد می پردازیم.

۲-۲- میکرو Atmega128 :

آیسی Atmega128 از سری میکرو کنترلر های ATMEGA شرکت ATMEL می باشد که کم مصرف است و از تکنولوژی CMOS در ساخت آن استفاده شده است. CPU این میکرو کنترلر ها ۸ بیتی هستند. مقدار ماشین سیکل آن ها تا 16 MIPS در فرکانس ١٦ مگاهرتز می رسد.این میکرو کنترلر ها قابلیت های بسیار زیادی دارند. شامل ۸ عدد مبدل آنالوگ به دیجیتال ۱۰ بیتی ، ۱۲۸ کیلوبایت حافظه Flash و ٤ کیلوبایت حافظه EEPROM[2] و ٤ کیلوبایت حافظه SRAM و تایمر / کانتر ۸ بیتی

[1] بزر

[2] Electrically Erasable Programmable Read-Only Memory

و ۱۶ بیتی ، پورت JTAG [1] و ۸ کانال PWM و بسیاری قابلیت های دیگر که این میکرو کنترلر تقریبا ارزان قیمت را برای پروژه های تجاری مناسب می کند.

شکل (۱-۲) آیسی ATmega128

۲-۳- LCD با کنترلر LM016L :

دو نوع LCD وجود دارد LCDهای کاراکتر و اعداد (متن) و LCD های گرافیکی.
LCD ۲×۱۶ یک LCD متنی است و دارای دو سطراست که هرسطر دارای 16 مکان برای نمایش کاراکتر می باشد.

شکل(۲-۲) LCD کاراکتری

LCD 2x16 دارای 16 پایه می باشد (در شکل فوق پایه های 1 تا 16 به ترتیب از چپ به راست قرار دارند).

[1] Joint Test Acsess Group

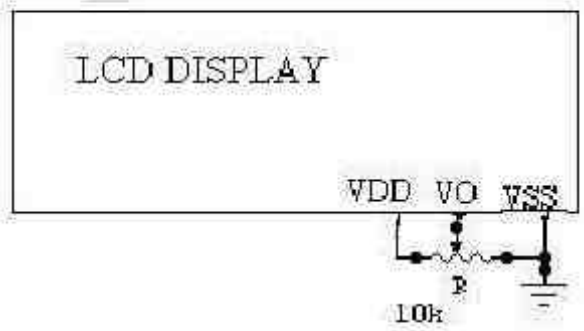
جدول (۱-۲) معرفی پایه های LCD

شماره پایه	سمبول	توضیحات
1	VSS	زمین منبع تغذیه
2	VDD	ولتاژ5+ ولت منبع تغذیه
3	V0	ولتاژ کنترل کنتراست
4	RS	اگر RS=0 باشد ثبات دستور انتخاب می شود و اگر RS=1 باشد ثبات داده انتخاب می شود.
5	R/W	R/W=0 برای نوشتن در LCD R/W=1 برای خواندن از LCD
6	E	فعال ساز

بیت های 0 تا 7 دیتا	D0 – D7	7-14
آنود لامپ LED پشت LCD	-	15
کاتود لامپ LED پشت LCD	-	16

ولتاژهای VSS و VDD تغذیه ی LCD را فراهم می کنند.

ولتاژ VO ولتاژ کنتراست است که تنظیم میزان روشنایی کاراکترها را روی LCD به کمک ولتاژهای VSS و VDD و یک مقاومت متغیر 10K . انجام می دهد (شکل زیر).



شکل (۳-۲) پتانسیومتر تنظیم کنتراست

در داخل LCD دو ثبات وجود دارد که توسط پایه RS انتخاب می شود.

اگر RS=0 باشد ثبات دستور IR انتخاب می شود تا اطلاعات ورودی به عنوان فرمان مشخص شوند. LCD این اطلاعات را دریافت می کند و فرمان تعریف شده را اجرا می کند . لیستی از این دستورات در جدول زیر موجود است.

در صورتیکه RS = 1 باشد ثبات داده DR انتخاب می شودتا کاربر بتواند اطلاعاتی را روی LCD بنویسد یا بخواند.

اطلاعات مربوط به کاراکتر ها باید به صورت کد اسکی باشد.

جدول دستورات:

جدول (۲-۲) تنظیمات LCD

عملکرد فرمان	کد هگزادسیمال فرمان
صفحه نمایش پاک می شود	1
مکان نما به محل اولیه بر می گردد	2
مکان نما پس از نوشتن هر حرف یا عدد به چپ شیفت پیدا می کند مکان نما پس از نوشتن هر حرف یا عدد به راست شیفت پیدا می کند	4 6

5	کاراکترها به راست شیفت پیدا می کنند
7	کاراکترها به چپ شیفت پیدا می کنند
8	کاراکترها و مکان نما خاموش می شوند
0A	
0C	کاراکترها خاموش و مکان نمای زیر خط ثابت روشن می شود
0D	کاراکترها روشن و مکان نما خاموش می شود
10	مکان نمای چشمک زن فعال می شود
14	
18	مکان نما به چپ شیفت پیدا می کند
1C	مکان نما به راست شیفت پیدا می کند
80	کل به چپ شیفت پیدا می کند
C0	کل به راست شیفت پیدا می کند
38	آدرس اولین کاراکتر سطر اول
	آدرس اولین کاراکتر سطر دوم

LCD به صورت دو سطری می شود	

پایه پنجم پایه خواندن یا نوشتن است. برای نوشتن روی LCD، باید R/W=0 باشد و برای خواندن اطلاعات از

LCD باید R/W = 1 باشد.

پایه 6 پایه فعال کردن (E) است. اگر در پایه E پالسی از 1 به 0 قرار داده شود، در این صورت اطلاعاتی که در پایه های 7 تا 14 قرار دارند در ثبات های LCD ذخیره می شوند. به عبارت دیگر در لبه منفی پالس ورودی به پایه E اطلاعات به LCD منتقل می شود.

پایه های 7 تا 14، 8 بیت اطلاعات ارسالی به LCD و یا دریافتی از آن می باشند. کد باینری دستورات و کد اسکی کاراکترها روی این پایه ها قرار می گیرند پایه های 15 و 16 برای لامپ پشت LCD می باشند.

جدول (۳-۲) آدرس های مکان کاراکتر های LCD

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

آدرس مکان کاراکترهای LCD به صورت هگزا دسیمال درجدول بالا مشخص گردیده است.

چگونگی استفاده از LCD :

از زمانی که پاور (منبع تغذیه) LCD وصل می شود تا زمان ارسال اولین اطلاعات روی LCD باید حدود 15 الی 20 میلی ثانیه تاخیر وجود داشته باشد.

قبل از استفاده از **LCD** باید به **LCD** مقدار اولیه داد. یعنی در موقع روشن کردن LCD و پس از تاخیر یاد شده باید موارد زیر را مشخص کنیم:

LCD به صورت دو سطر یا یک سطر عمل کند .

با فرمان 38h , LCD دو سطری می شود که بهتر است درحالت دو سطری باشد.

وضعیت مکان نما و کاراکترها مشخص شود.

یعنی اینکه مکان نما و کاراکترها هر دو روشن یا یکی خاموش و دیگری روشن باشد و یا مکان نما چشمک زن باشد.با یکی از دستورات 0Ah,0Ch,0Dh می توان و ضعیت اولیه مکان نما و کاراکتر ها را مشخص نمود. (پس از دادن مقدار اولیه دادن باز هم می توان به طور دلخواه وضعیت مکان نما و کاراکترها را تغییر داد)

با دستور 1h صفحه نمایش پاک می شود.

کاراکترها ثابت بمانند و مکان نما پس از نوشتن هر حرف به چپ یا راست برود و یا مکان نما ثابت باشد و کاراکترها تغییر وضعیت دهند.

حتما باید مقدار دهی اولیه انجام شود, در غیر اینصورت LCD کار نخواهد کرد.

آخرین نکته ای که باید ذکر شود این است که چون بعد از هر فرمان و دیتایی که به LCD ارسال می شود مدت زمانی لازم است تا LCD این اطلاعات را دریافت کند لذا برای ارسال دستور بعدی باید چند میلی ثانیه تاخیر داشت . برای LCD های موجود در بازار ایران بهتر است که این زمان 5 میلی ثانیه باشد.

جدول (٤-٢) کد کاراکتر های LCD

۲-٤- آشنایی با GSM[1]:

GSM MODEM چیست؟

در واقع GSM MODEM دستگاهی است که امکاناتی همانند یک تلفن همراه را در اختیار ما قرار می دهد به وسیله GSM MODM می توان پیامک فرستاد ، از سرویس اینترنت همراه (GPRS) استفاده کرد و حتی می توان به یک شماره تلفن زنگ زد و مکالمه کرد.

شرکت های زیادی در دنیا انواع GSM MODEM ها را برای کاربرد های مختلف عرضه نموده اند برخی از آن ها برای اتصال مستقیم به پورت USB کامپیوتر طراحی شده اند و برخی دیگر برای کاربرد صنعتی و عمومی تر.

از جمله موارد استفاده از GSM Modem استفاده از آن در دزدگیر ها است که به محض فعال شدن دزدگیر با صاحب محل و پلیس تماس گرفته و پیامک ارسال می کند و حتی می توان بوسیله پیامک دزدگیر را خاموش و روشن کرد و با آسانسور هایی که در صورت خرابی به صورت خودکار با شرکت سرویس دهنده آسانسور تماس می گیرند و آن ها را مطلع می کنند و ...

۲-٤-۱- انتخاب یک GSM Modem مناسب:

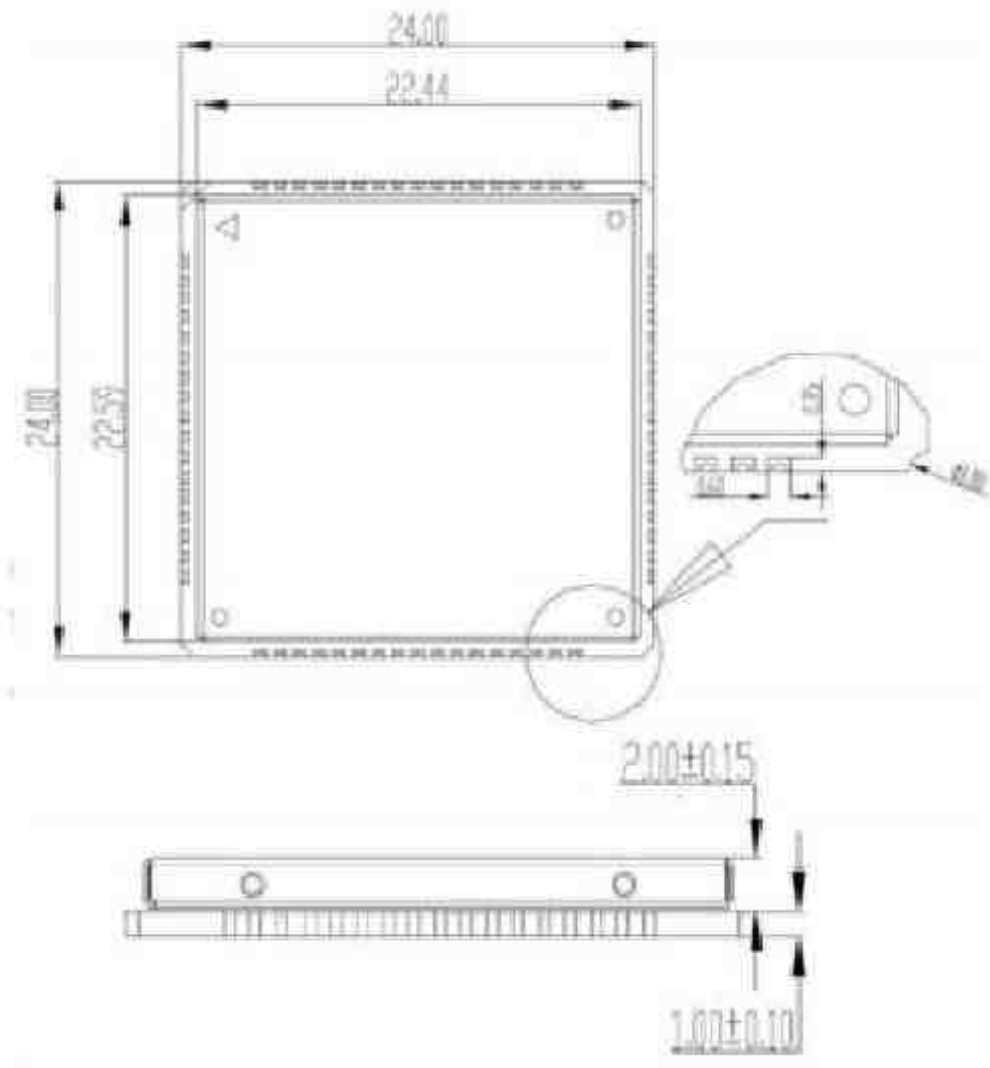
بسیاری از GSM Modem ها بوسیله پورت سریال دستورات مورد نظر را دریافت و تقریبا تمام میکروکنترلرهای دنیا پورت سریال (UART[2]) را پشتیبانی می کنند.

در اینجا GSM Modem انتخابی ما برای کار ماژول SIM900 استکه علاوه بر فراوانی در بازار ایران، قیمت مناسب و کارایی بالایی دارد. ابعاد این مودم بسیار کوچک است که استفاده از آن را در

[1] Global System for Mobile communication
[2] universal asynchronous receiver/transmitter

دستگاه ها به مراتب ساده تر می کند.

در حال حاضر برای سیستم های میکروکنترلری بهترین گزینه استفاده از SIM900 است.



شکل (٤-٢) پکیج ماژول GSM ، SIM900

۲-٤-۲- آشنایی با سخت افزار SIM900

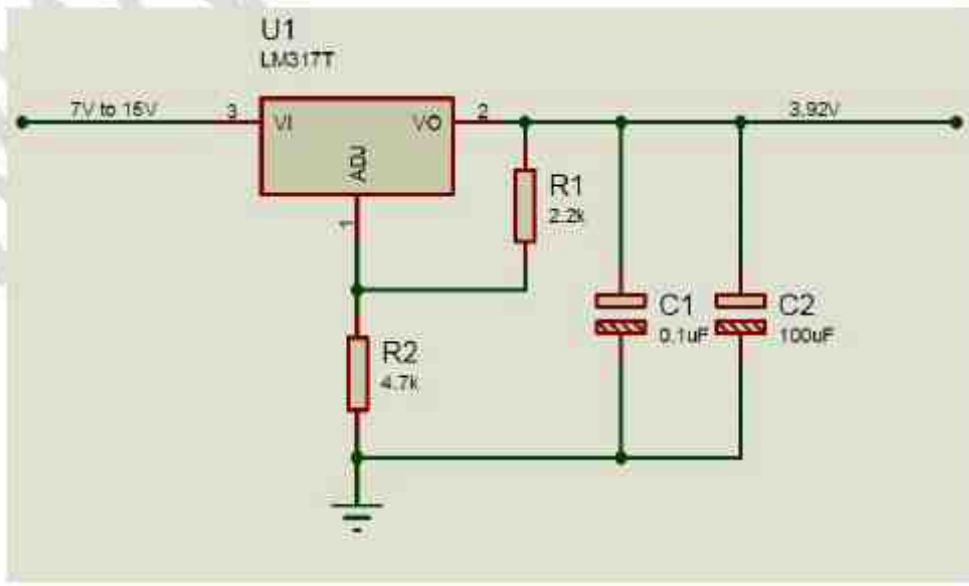
الف) منبع تغذیه برای SIM900

این ماژول برای کار نیاز به یک منبع با ولتاژ بین 3.4V تا 4.5V دارد. همچنین مقدار متوسط جریان مصرفی آن در حالت بیکاری 22mA و در حالت کار (مانند ارسال پیامک) حداکثر 400mA است.

نکته مهمی که در اینجا وجود دارد پیک بالای جریان لحظه ای این ماژول است. در شکل زیر، شکل موج جریان این ماژول را می توان مشاهده کرد. مطابق این شکل موج ماژول SIM900 به صورت لحظه ای جریانی تا حدود 2A مصرف می کند که در پیک جریان ولتاژ تغذیه نباید بیش از 400mV افت کند. در غیر این صورت ماژول ریست می شود.

برای تامین این پیک جریان استفاده از دو خازن تانتالیوم موازی با ظرفیت 100uF و 100nF در مسیر تغذیه ماژول الزامی است.

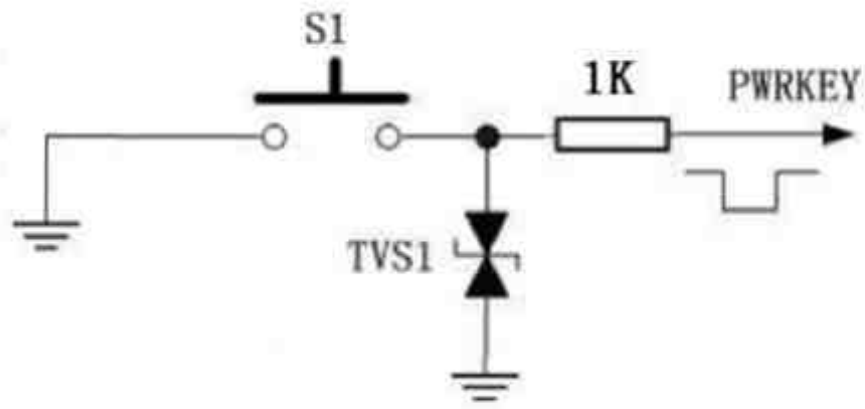
همچنین برای مدار رگولاتور می توان از مدار زیر استفاده کرد.



شکل (۲-۵) رگولاتور LM317T تغذیه GSM

۲-٤-۳ روشن کردن SIM900:

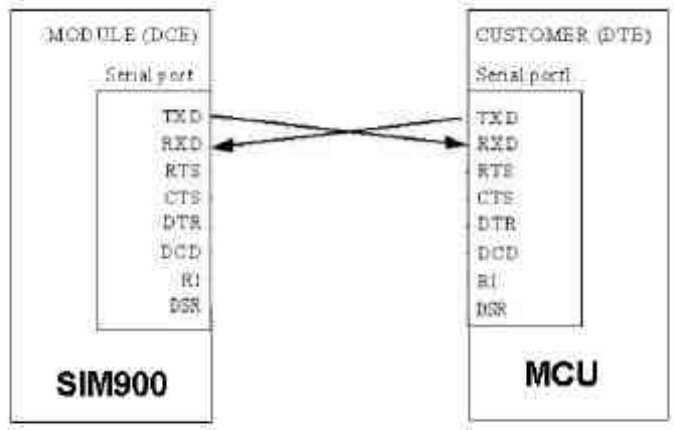
پس از اعمال ولتاژ به این ماژول برای روشن کردن آن بایستی پایه PWRKEY را برای مدت حداقل یک ثانیه صفر و سپس دوباره یک شود. برای خاموش شدن ماژول نیز باید همین طور عمل کنیم.



شکل (۲-٦) سوییچ پاور ماژول SIM900

۲-٤-٤-اتصال SIM900 به میکروکنترلر

همانطور که گفته شد ماژول SIM900 از طریق پورت سریال (UART) ارتباط برقرار می کند. در شکل زیر حداقل سیم بندی مورد نیاز برای ارتباط این ماژول با میکروکنترلر را مشاهده می کنید.



شکل (۲-۷) نحوه اتصال ماژول به میکروکنترلر با پورت سریال

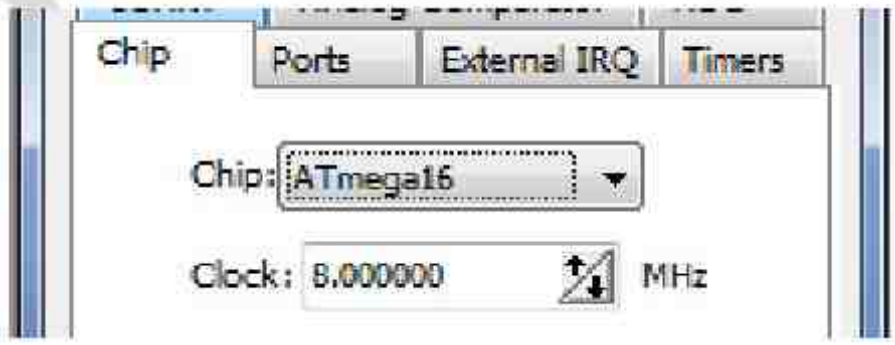
پروتکل ارتباط سریال

UART دارای چندین مشخصه است:

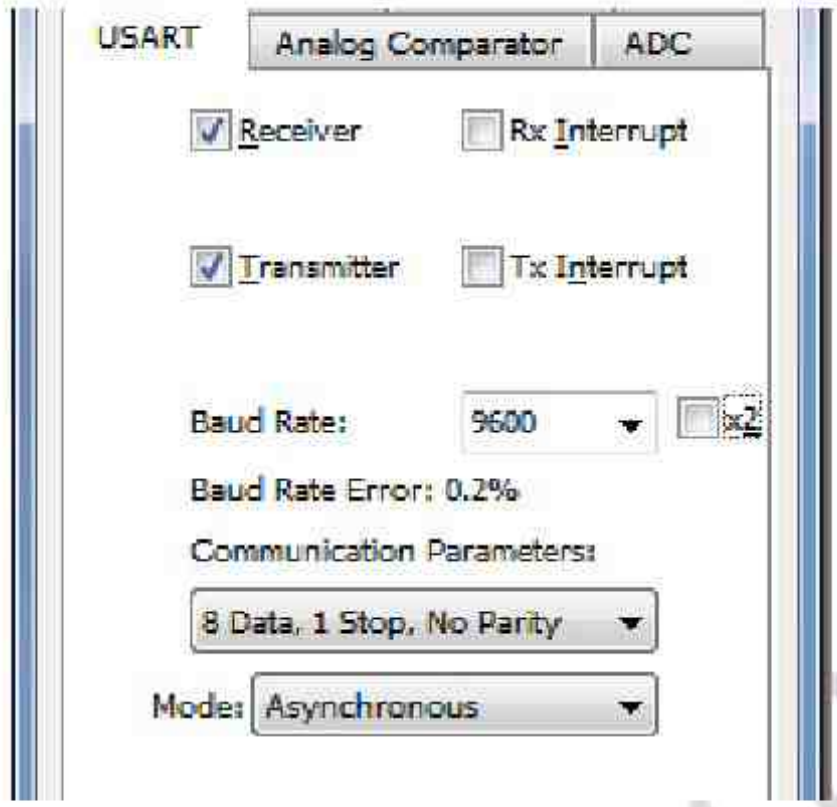
- تعداد بیت های هر فریم که در اینجا 8 bit است.
- تعداد Stop bit ها که در اینجا 1 است.
- نوع Parity که در اینجا غیر فعال است.
- سرعت ارتباط که می تواند هر یک از مقادیر 1200bps , 2400bps , 4800bps , 9600bps , 19200bps , 38400bps , 57600bps , 115200bps باشد. ماژول سرعت ارتباط را به طور اتوماتیک تشخیص می دهد.

همانطور که گفته شد ماژول SIM900 این قابلیت را دارد که سرعت ارتباط را به طور خودکار تشخیص دهد ، برای این کار ابتدا باید در ابتدا یک کاراکتر 'A' به ماژول ارسال کنیم.

۲-٤-٥- ساخت یک پروژه

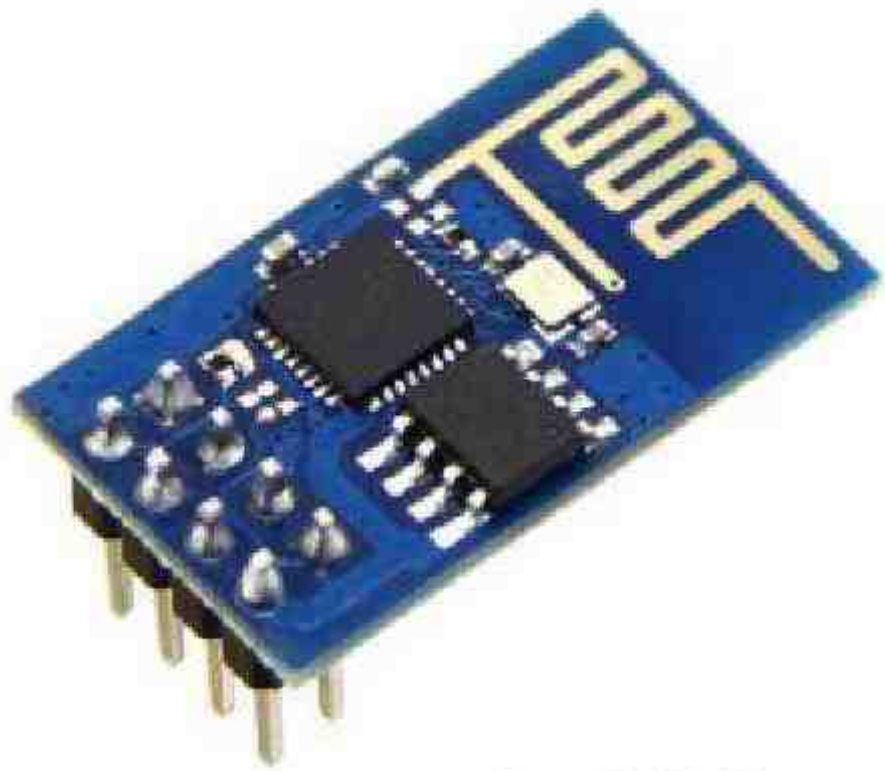


شکل (۸-۲) انتخاب میکرو در ویزارد کدویژن



شکل (۹-۲) تنظیمات مربوط به USART

۲-۵-۱- آشنایی با ماژول وای فای ESP8266:



شکل (۱۰-۲) ماژول وای فای ESP8266

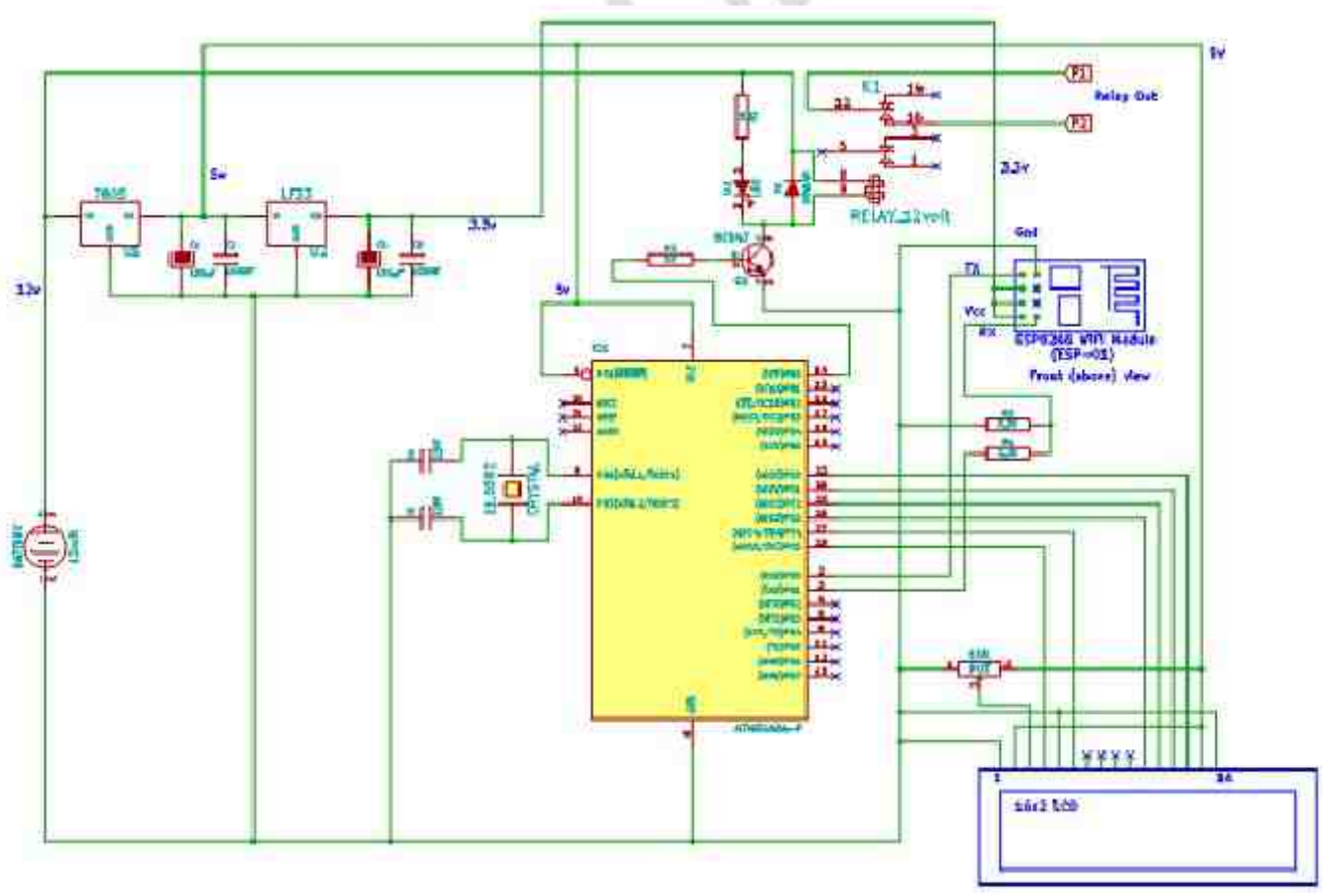
ماژول WiFi وای فای ESP8266 در مد AP قادر است همانند یک سرور کوچک عمل کرده و ضمن اینکه می توان برای این اکسس پوینت (همانند مودم ADSL) نام SSID و Password در نظر گرفت می توان با AT Command خاصی ماژول را وادار به گوش دادن به پورت خاص نمود. شایان ذکر است که ماژول ESP8266 در مد AP معمولاً همیشه آی پی ثابت و فیکس ۱۹۲.۱۶۸.٤.۱ را گرفته و با تعریف یک پورت دلخواه مثلا پورت شماره ۱۳۹٤ می توان از ترکیب این آی پی و پورت یک سوکت مانند (۱۹۲.۱۶۸.٤.۱:۱۳۹٤) ایجاد کرد و با استفاده از نرم افزارهای مختلف تحت اندروید یا Windows همانند Telnet به این ماژول متصل شد و رشته های متنی را برای آن ارسال کرد. این رشته ها متقابلاً در خروجی پایه TX ماژول قابل دریافت بوده و می توان با کمک آن رله ای را فعال یا غیر فعال کرد.

از این ایده می توان برای ساخت ریموت کنترل WiFi وای فای بدون نیاز به اینترنت و روتر و مودم استفاده کرد. در این ایده با استفاده از یک موبایل یا تبلت یا لپ تاپ دارای WiFi براحتی میتوان نام ماژول را جستجو کرده و پس از وارد کردن رمز عبور به ماژول متصل شد. پس از متصل شدن به

ماژول بسته به نوع دستگاه ما (اندروید، ویندوز، IOS و ...) بایستی به نحوی به سوکت ماژول متصل شده و رشته دلخواه مثلا on یا Off را برای ماژول ارسال کنیم. برای این منظور ما در گوشی اندروید از نرم افزار رایگان تل نت Telnet استفاده کرده ایم که می توانید از انتهای همین صفحه فایل APK آن را دانلود نمایید. برای ویندوز فون هم مطمئناً این نرم افزار در دسترس خواهد بود و در بر روی تبلت و لپ تاپ ویندوز هم بصورت پیش فرض Telnet نصب می باشد.

برنامه ای که برای میکروکنترلر ATmega8 با کامپایلر BASCOM بسکام/بیسکام نوشته شده یک ریموت کنترل یک کانال وای فای WiFi با ماژول ESP8266 می باشد و قادر است در پاسخ به رشته های on و off رله خود را روشن و خاموش کرده و پیام Delivery برای دستگاه ارسال نماید. همچنین با ارسال رشته status وضعیت کنونی رله را ارسال می نماید. توجه داشته باشید که رشته ها کاملاً با حروف کوچک و بدون فاصله اضافه در ابتدا و انتها در Telnet تایپ شود.

۲-۵-۲- شماتیک اتصال ماژول به میکرو:



شکل (۱۱-۲) شماتیک اتصال ماژول وای فای به میکروکنترلر AVR

شماتیک بالا برای اتصال ماژول ESP8266 به میکرو AVR می باشد. در این شماتیک ماژول اطلاعات دریافتی را با استفاده از پورت سریال خود به پورت سریال میکرو می فرستد ، سپس از آن جا اطلاعاتی استخراج شده بر روی LCD کاراکتری به نمایش در می آید.

۲-۵-۳- نحوه راه اندازی :

پس از پروگرام کردن فایل Hex در میکروکنترلر ATmega128 و تنظیم فیوز بیت اسیلاتور روی کریستال خارجی (۱۱۱۱) و اتصال تغذیه مشاهده می شود که عباراتی برای تنظیم اولیه ماژول بر روی LCD نوشته می شود. پس از حدود ۵ ثانیه عبارت Wifi Ready بر روی LCD نمایش داده می شود. این به معنی آماده بودن دستگاه برای اتصال است.

در گوشی تلفن همراه وارد قسمت تنظیمات وای فای شده. بایستی دستگاهی به نام FATTAHI را مشاهده گردد. دستگاه مذکور که همین ماژول ESP8266 می باشد را انتخاب کرده و برای اتصال، گزینه رمزنگاری WPA/WPA2 را انتخاب و رمز گذاشته شده بر روی ماژول را وارد کنید تا به ماژول متصل شویم. در صورت اتصال موفقیت آمیز علامت آنتن وای فای در گوشی شما روشن شده و سیگنال با قدرت بالا را نمایش می دهد (قدرت ماژول ESP8266 از مودم های خانگی به مراتب بیشتر است).

۲-۵-٤- نکاتی درباره ESP8266 :

از جمله نکات مهم در خصوص این پروژه نوع ماژول های ESP8266 موجود در بازار است. برخی از این ماژول ها با سرعت ۱۱۵۲۰۰ بیت در ثانیه کار میکنند و بعضی دیگر با سرعت ۹۶۰۰ که بسته به ماژول ممکن است با یکی از این فایلها قادر به ارتباط با ماژول باشید. نکته دیگر ولتاژ کاری ماژول است که ۳.۳ ولت بوده و میکرو هم با ۵ ولت راه اندازی شده و رله نیز با ۱۲ ولت درایو شده است. این موارد را از روی شماتیک می توانید بررسی کنید. نکته مهم دیگر تنظیم فیوزبیت نوسان ساز میکرو

بر روی کریستال خارجی می باشد که در غیر این صورت میکرو قادر به ارتباط با ماژول نخواهد بود. همچنین برای صحت کار ماژول می توانید پس از بستن شماتیک این پروژه و پروگرم کردن میکرو، محیط ترمینال را باز کرده تنظیمات را بر روی سرعت = ۱۱۵۲۰۰ یا ۹۶۰۰، دیتابیت = ۸، پریتی n = ، و استاپ بیت = ۱ قرار داده و پایه TX ماژول را که به RX میکرو متصل شده با یک سیم به پایه RX پورت سریال کامپیوتر هم متصل کنید (زمین پورت سریال کامپیوتر هم به زمین مدار وصل شود) تا اطلاعات ارسالی ماژول را در محیط ترمینال نیز ببینید و از صحت کار ماژول مطمئن شد.

۲-۵-۵- استفاده از ESP8266 برای کنترل اشیاء:

اینترنت اشیاء یکی از پدیده های جدید در سالهای اخیر (۲۰۱٤ و ۲۰۱۵) به شمار می رود که در نمایشگاه CES2015 نمونه های فراوانی از آن به چشم می خورد. و منظور از آن اینست که هر شیئی در دنیای فیزیکی به شبکه جهانی اینترنت متصل شود یا تمام اشیاء با یکدیگر شبکه شده و به اینترنت متصل شوند و در نهایت بتوان تمام اشیاء را از طریق اینترنت کنترل و مانیتور کرد. منظور از اشیاء، دستگاه های الکتریکی و الکترونیکی مانند سیستم تهیه هوا، پنکه، لامپ، سنسور دما، دزدگیر، درب منزل و غیره می باشد. با این ایده تمام سیم کشی های اضافی و شبکه های رنگارنگ خاص که برای سیستم های حفاظتی، اعلام حریق، درب بازکن منزل، تلفن داخلی، روشنایی ساختمان و غیره کشیده می شود حذف شده و تمام این ها از طریق اینترنت به یکدیگر متصل می شوند (تقریباً می توان گفت تمام وسیله ها مثل دوربین های جدید IP به اینترنت متصل شده و از طریق اینترنت کنترل می شوند).

ممکن است تصور شود برای متصل کردن هر وسیله ساده مثل لامپ به اینترنت نیاز به کابل شبکه و مودم و یک میکرو کامپیوتر داریم که کار را مشکل و گران می کند؛ اما شرکت های مختلف ماژول های کوچکی را برای این منظور طراحی کرده اند. یکی از این ماژول ها که ESP8266 نامیده می شود از یک آی سی با همین نام استفاده می کند و در واقع یک ماژول وای فای(WiFi)کامل به همراه تمام بخش های نرم افزاری و پشته پروتکل داخلی TCP/IP می باشد که با قیمت کم (در حدود ۵$) ارائه می شود و با پروتکل سریال (رابط RS232 یا TTL) می توان آن را به یک میکروکنترلر کوچک AVR متصل کرد.

۲-۵-٦- حالت های استفاده ESP8266 :

ماژول ESP8266 در دو حالت قابل استفاده است. حالت اول که ما از آن برای اینترنت اشیاء استفاده کرده ایم حالت Station می باشد که پس از تنظیمات لازم بصورت خودکار مودم ADSL را پیدا کرده و خود را به شبکه جهانی اینترنت متصل می کند. حالت دوم هم مد Access Point است که می توان با موبایل یا تبلت ماژول را جستجو کرد و به آن متصل شد و عموماً برای ساخت ریموت های تنها و بدون نیاز به اینترنت بکار می رود.

۲-۵-۷- نکاتی درباره برنامه نویسی میکروجهت راه اندازی ESP8266 :

۱- ماژول ESP8266 به سایز حروف حساس می باشد حتماً تمام AT Command ها را با حرف بزرگ ارسال کنید.

۲- ماژول ESP8266 فوق العاده سریع بوده و ممکن است با برنامه های ترمینال قادر به دریافت اطلاعات ماژول نباشید.

۳- برای راه اندازی ماژول با Hyper terminal مجبور به تنظیم تاخیر خط و تاخیر کاراکتر و نوع کاراکتر پایان خط CR و LF شدیم.

٤- بعد از Hyper با دو نرم افزار ترمینال دیگر نیز قادر به ارتباط با ماژول نشدیم.

۵- آخرین انتخاب، برنامه ترمینال کامپایلر آردوینو بود که تنظیم کاراکتر پایان خط آن روی CR and LF قرار داشت.

٦- در ارتباط با سرور TCP پس از Link شدن فقط ۹ ثانیه فرصت دارید که HTTP GET Request را تایپ شود که غیر ممکن است!

۷- برای ارسال درخواست Http Get به سرور باید تعداد کاراکترهای ارسالی را قبل از تایپ شمرده و به طور دقیق به سمت سرور ارسال کنید!

(احتمالا به این دلیل است که سایز packet های IP مشخص و بزرگ بوده و برای تعیین دقیق سایز درخواست باید به سرور بگوییم که طول درخواست ما چقدر است)

۸- نظر به اینکه وبسایت های ارزان قیمت از سرور اشتراکی استفاده میکنند و ممکن است صدها وبسایت یک IP داشته باشند بایستی نام دامین را داخل درخواست GET بنویسید.

(چون DNS نام دامنه شما را به IP تبدیل کرده و به صفحه ادمین ٤۰٤ وب هاست منتقل می شوید

ولی با نوشتن نام Domain به پوشه اشتراکی سایت هدایت خواهید شد)

۹- تعداد کاراکترهای ارسالی از شمردن تعداد کارکترهای درخواست HTTP همراه با فضای سفید به علاوه عدد ٦ بدست می آید، عدد ٦ همان سه جفت کاراکتر CR-LF پایان سه خط دستور HTTP GET می باشد.

۱۰- خط سوم درخواست فقط شامل CR-LF می باشد. به عبارت دیگر پس از تایپ HTTP GET بایستی جفت Enter کنید تا درخواست به سمت سرور ارسال شود!

تمام نکات بالا در برنامه نوشته شده با کدویژن رعایت شده. برای نوشتن این برنامه مجبور به مطالعه کتب تخصصی شبکه و اطلاعاتی در خصوص نحوه عملکرد web server ها و سرورهای اشتراکی و درخواست های HTTP و متدهای GET و POST و غیره شدیم و به همین دلیل امیدواریم دلیل پولی بودن سورس این پروژه را درک کنید و با خرید و تعهد بابت عدم پخش آن در اینترنت از برنامه نویس حمایت نمایید.

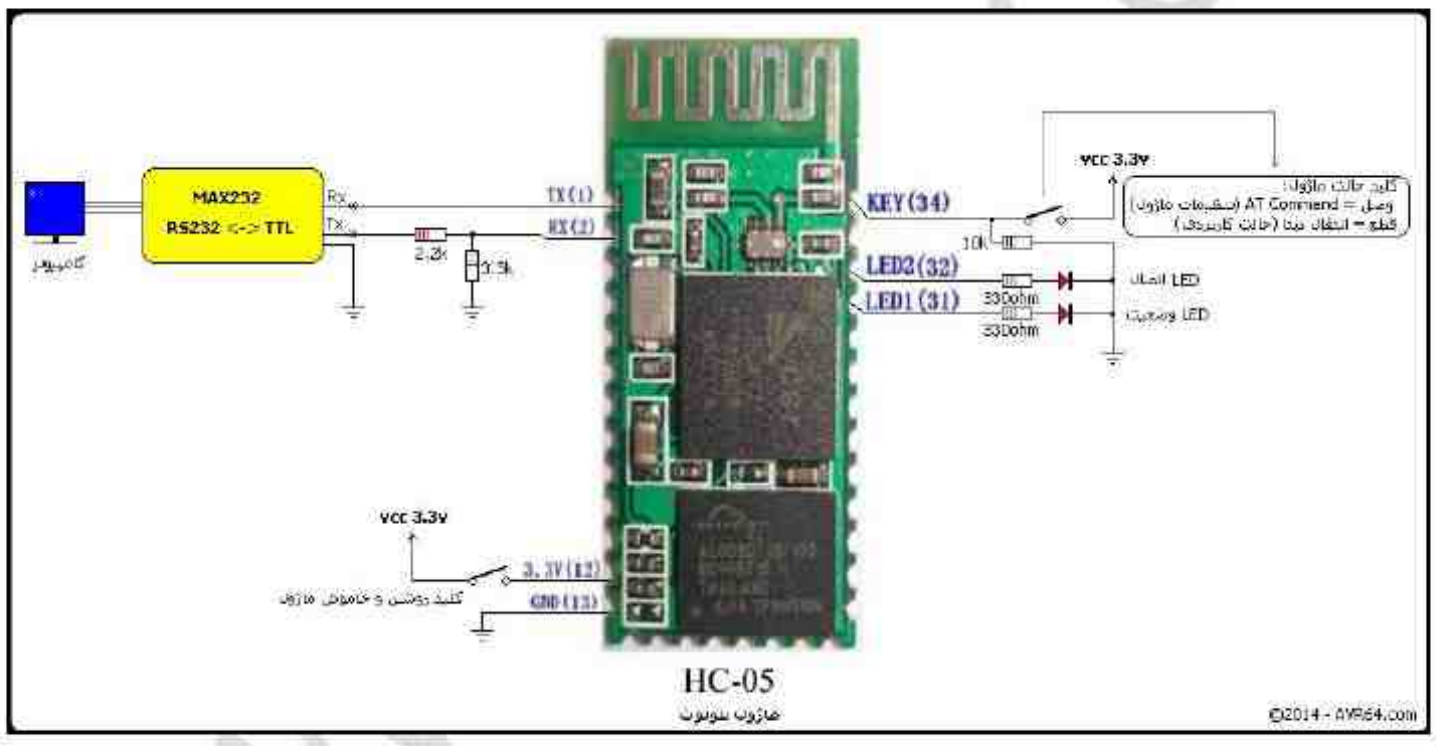
۲-۶-۱- معرفی ماژول بلوتوث HC-05 :

شکل (۱۲-۲) ماژول بلوتوث HC-05

ماژول بلوتوث HC-05 یک ماژول استاندارد Bluetooth با خروجی سریال می باشد. این ماژول برای ارتباط بین میکروکنترلر و موبایل و تبلت و نیز دو میکرو با یکدیگر با سرعت بالا انتخاب خوبی به شمار می رود. همچنین میتوان با استفاده از این ماژول بین پروژه های ساخته شده با میکرو و lap top های دارای پورت USB و PC های دارای دانگل بلوتوث نیز یک لینک ارتباطی بیسیم از نوع ارتباط سریال برقرار کرد. از جمله کاربردهای این ماژول می توان موارد زیر را برشمرد:

- ساخت روبات کنترل از راه دور با موبایل
- نمایش اطلاعات مربوط به یک پروژه دماسنج در یک بازه زمانی خاص بر روی تبلت یا موبایل و رسم نمودار
- ساخت پروگرمر بیسیم برای پروگرم کردن میکروکنترلر با استفاده از تبلت
- ریموت کنترل درب منزل با استفاده از موبایل
- اتوماسیون خانه و محیط کار با موبایل و تبلت

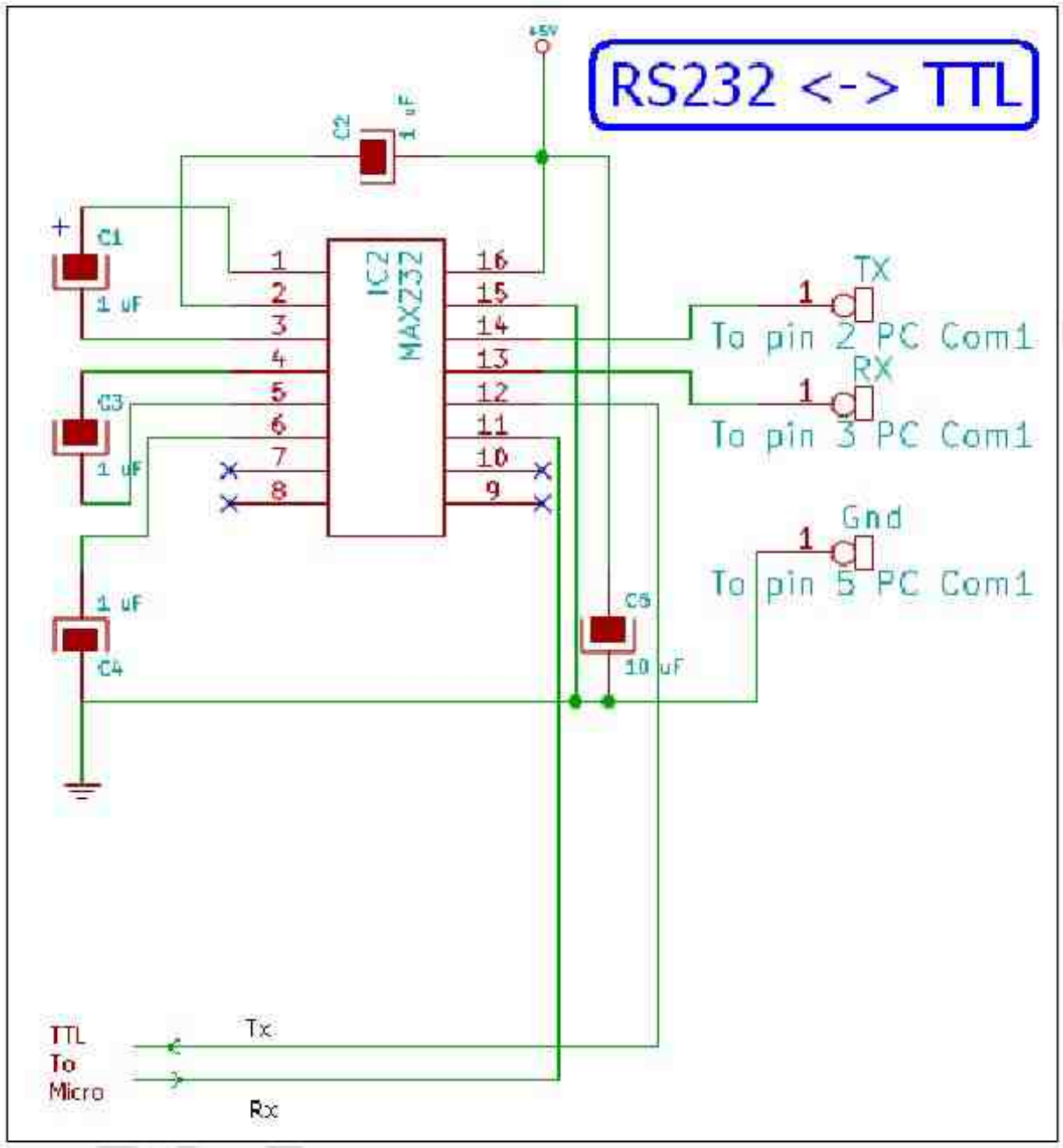
ماژول بلوتوث HC-05 قبل از راه اندازی بایستی پیکره بندی شود و این کار بوسیله اتصال این ماژول به یک ترمینال استاندارد سریال صورت می گیرد. برای این منظور بایستی ابتدا مداری مطابق شکل زیر تشکیل داده و ماژول را بوسیله یک رابط استاندارد RS232-TTL به پورت سریال یک کامپیوتر متصل نمایید و یا از ماژول USB به سریال با خروجی TTL برای اتصال این ماژول به کامپیوتر بهره بگیرید. توجه داشته باشید که ولتاژ کاری ماژول بلوتوث ۳.۳ ولت می باشد و برای همین منظور از شبکه تقسیم مقاومتی بر روی پایه TX ماژول سریال به سمت RX ماژول بلوتوث استفاده کرده ایم تا ولتاژ ۵ ولت به میزان ۳.۳ ولت کاسته شود. این شبکه بر روی پایه دیگر مورد نیاز نمی باشد چرا که پایه RX ماژول سریال فقط دریافت کننده می باشد و ولتاژ سیگنال ۳.۳ ولت می تواند به طور مستقیم وارد ماژول سریال گردد و یقیناً باعث تحریک ماژول سریال خواهد شد.



شکل (۱۳-۲) پایه های ماژول HC-05 و نحوه اتصال آن به رایانه

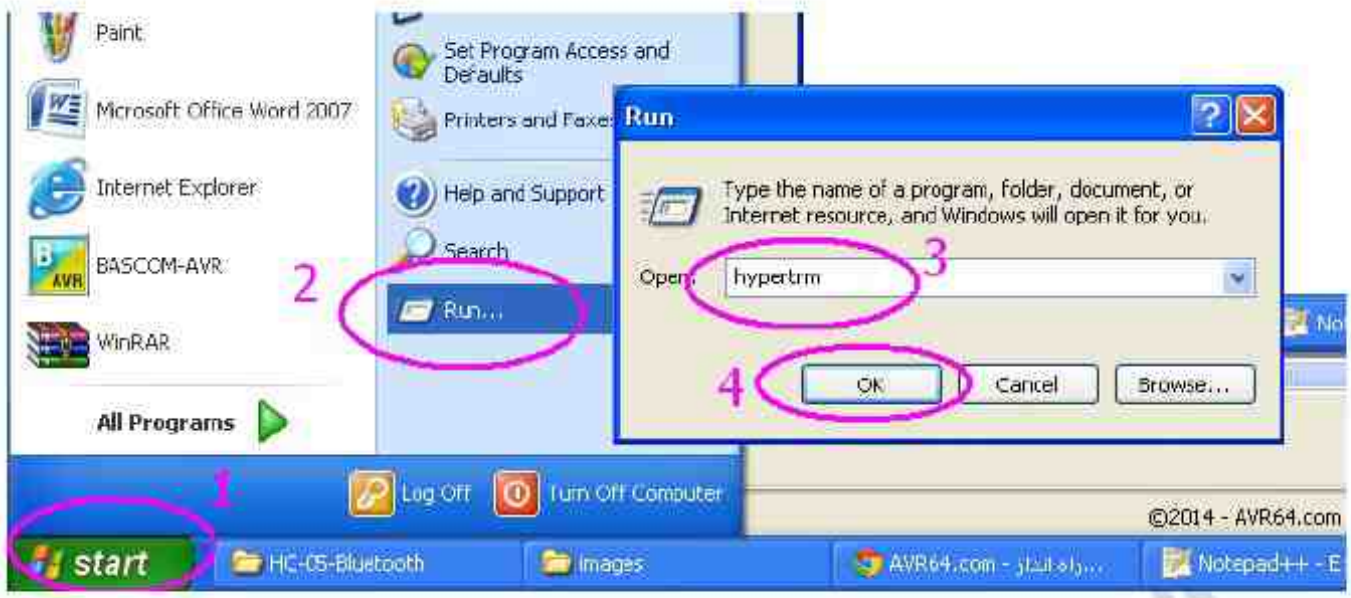
در شماتیک بالا بخش زرد رنگ ماژول مبدل RS232-TTL می باشد که براحتی با استفاده از آی سی MAX232 و چند خازن و یک پورت مادگی DB9 قابل ساخت است. شماتیک زیر نحوه ساخت این ماژول را نشان می دهد.

۲-٦-۲- راه اندازی ماژول HC-05:



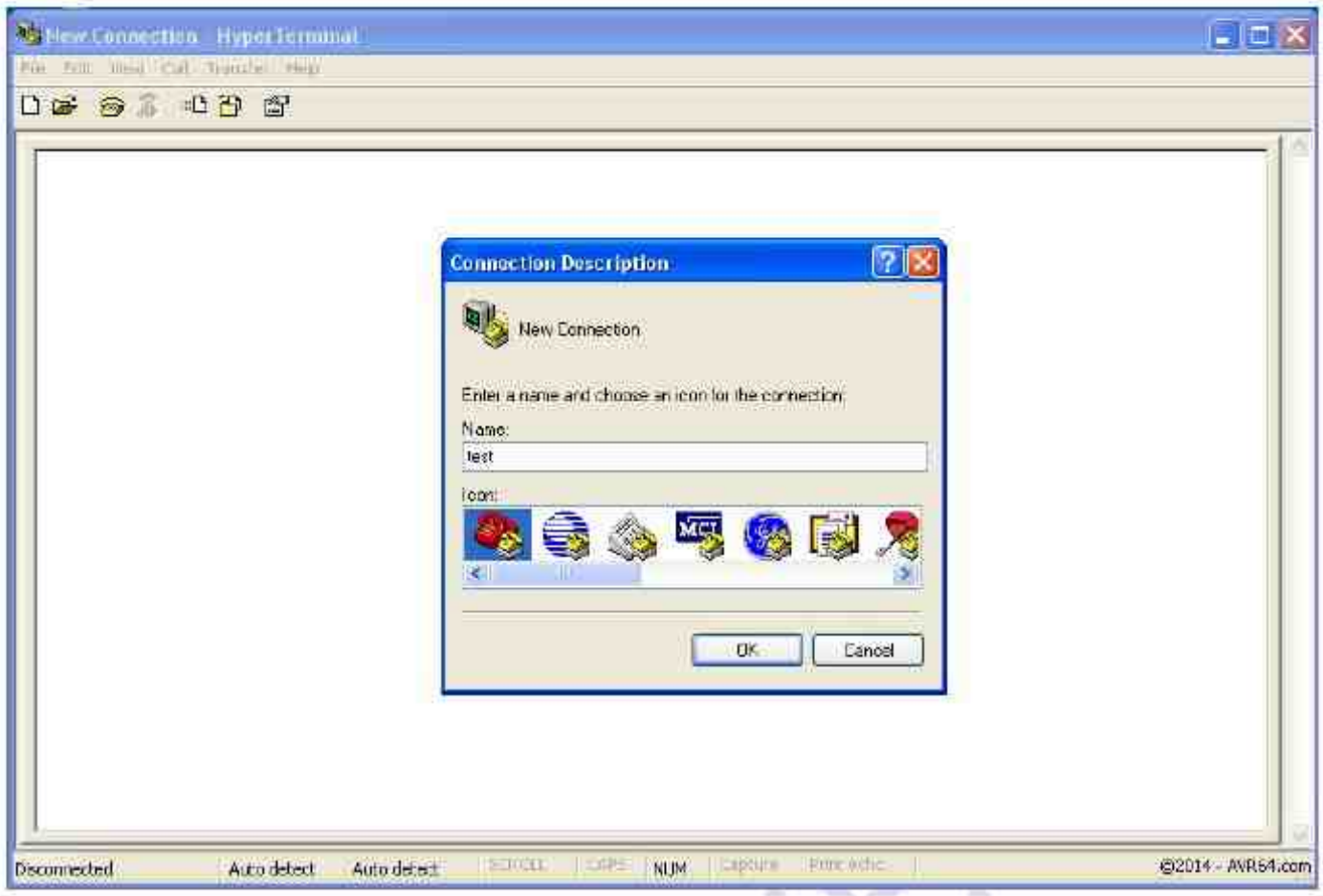
شکل (۱٤-۲) تبدیل TTL به پروتکل RS232

قبل از هر چیز در شماتیک بالا هر دو کلید تغذیه ماژول و حالت ماژول را در حالت قطع قرار دهید. سپس مانند شکل زیر از منوی **Start** بر روی **Run** کلیک کرده و در پنجره **Run** عبارت **hypertrm** را تایپ کرده و با کلید OK ارسال می گردد.



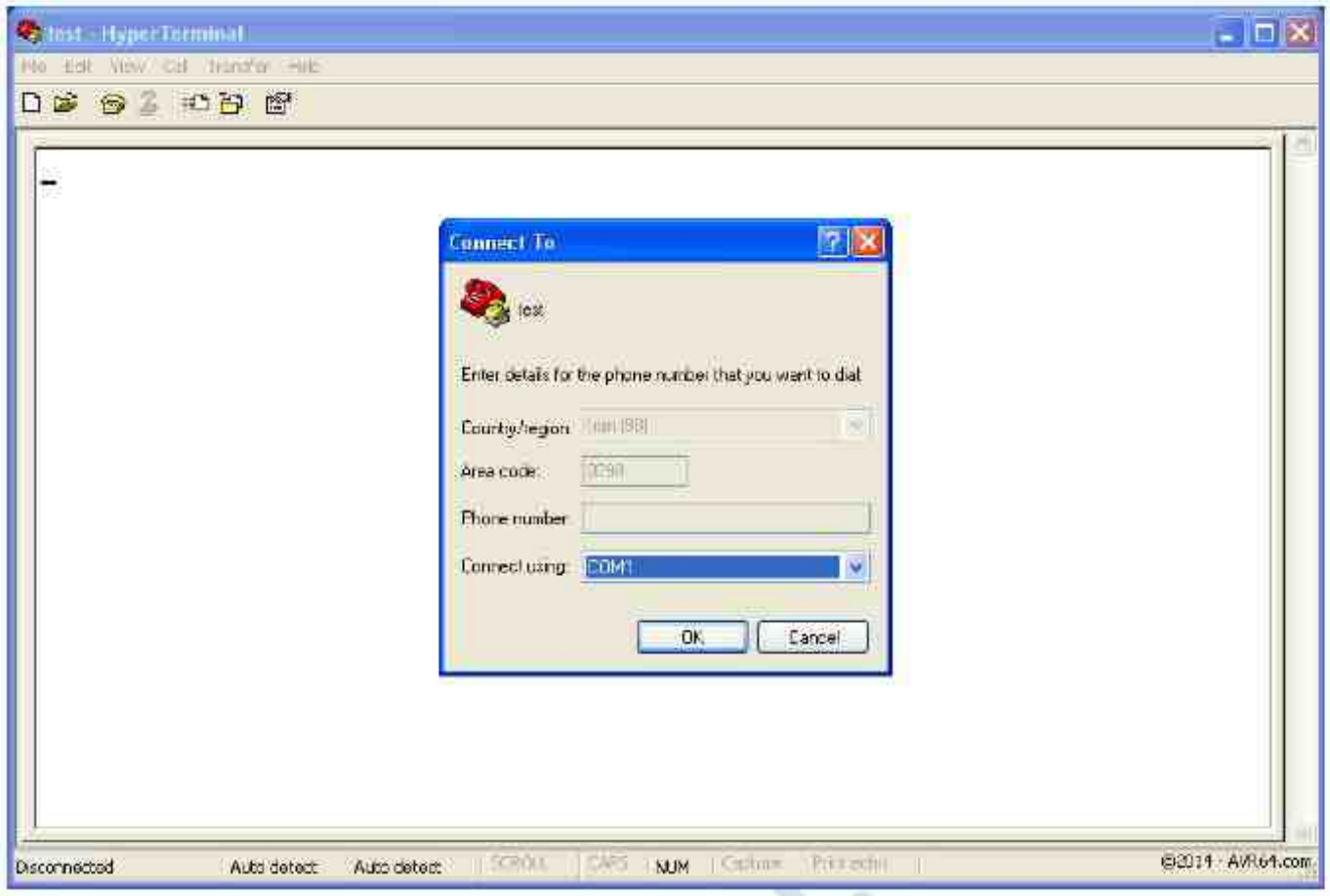
شکل (۱۵-۲) نحوه اجرای هایپر ترمینال در محیط ویندوز

اگر اولین بار است که محیط هایپر ترمینال ویندوز را باز می کنید ممکن است سوالاتی در خصوص تنظیم به حالت پیشفرض و انتخاب کشور و مودم پیشفرض و کد کشور (۰۰۹۸) پرسیده شود. پس از پاسخگویی به تمام سوالات پنجره ای مطابق شکل زیر نمایش داده می شود. یک نام دلخواه وارد کنید و بر روی کلید OK کلیک نمایید.



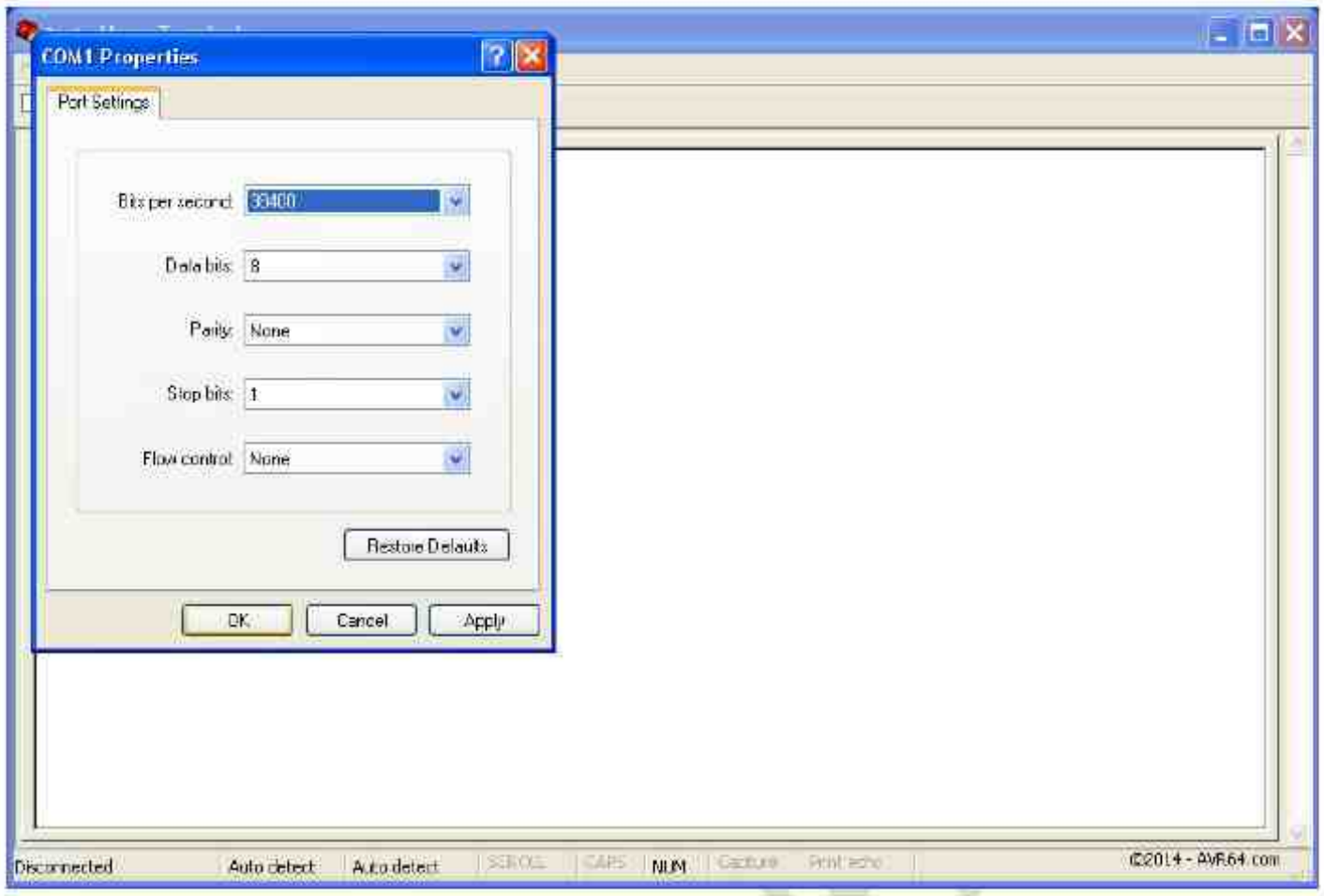
شکل (۱۶-۲) محیط هایپر ترمینال ویندوز

در قسمت بعد پنجره ای نمایش داده می شود و نام پورتی که ماژول RS232-TTL را به آن متصل کرده اید پرسیده می شود. اگر کامپیوتر شما دارای یک یا دو پورت سریال باشد و از ماژول RS232-TTL یا آی سی MAX232 معمولی استفاده کرده اید بایستی COM1 یا COM2 را انتخاب کرده و برای مطمئن شدن بهتر است قبلاً با یک پروژه سریال میکروکنترلری صحت ارتباط را چک کرده باشیم. اما اگر از ماژول USB به سریال استفاده شود بایستی نام پورت مجازی را بدانید که این کار با بررسی نام پورت مجازی شناخته شده در Device Manager کامپیوتر قابل انجام است. همچنین در صورت استفاده از ماژول USB به سریال بایستی توجه داشته باشید که برخی از ماژول ها خروجی TTL داشته و برخی دیگر خروجی RS232 دارند که سطح ولتاژ ومنطق آن متفاوت با TTL بوده و نیاز به آی سی MAX232 احساس می شود. (استفاده از نام های TTL و RS232 به این شکل صحیح نیست و RS232 در واقع نام رابط سریال است که به دلیل عمومی بودن و قابل فهم بودن مقاله به این شکل ذکر شده است).



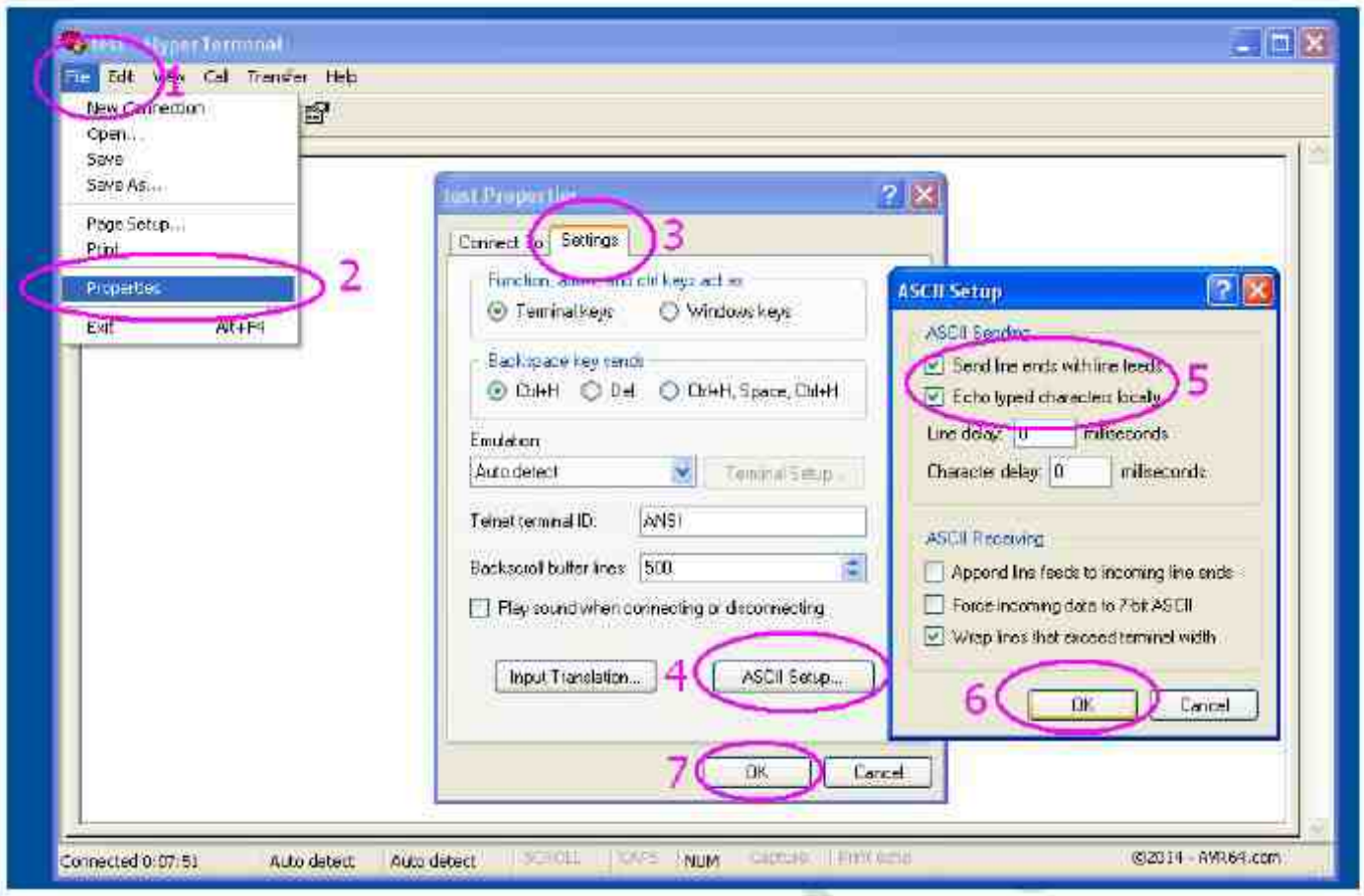
شکل (۱۷-۲) انتخاب پورت COM مورد نظر در محیط هایپر ترمینال

پس از تنظیم نام پورت وارد پنجره تنظیمات ارتباط سریال از قبیل سرعت پورت (باود ریت) و غیره می شوید. این تنظیمات را دقیقاً مطابق شکل زیر انجام دهید و برای باز کردن پورت روی OK کلیک نمایید.



شکل (۱۸-۲) تنظیمات پورت سریال در هایپرترمینال

هم اینک پورت سریال باز شده است و تایمر پایین پنجره این واقعیت را نشان می دهد. مرحله آخر تنظیم ارسال CRLF و Echo ی کارکترهای تایپ شده است. مطابق شماره های شکل زیر مراحل را طی کنید و در مرحله شماره ۵ تیک هر دو گزینه را بزنید.



شکل (۲-۱۹) تنظیمات تست هایپر ترمینال

حال بایستی مطابق دستور العمل زیر پیش بروید و ماژول بلوتوث را **Setup** کنید (این تنظیمات فقط یک بار لازم بوده و در حافظه ماژول باقی می ماند).

۱- کلید تغذیه ماژول قطع باشد.

۲- کلید حالت ماژول (متصل شده به پایه ۳٤ ماژول) را وصل کنید (اتصال پین ۳٤ ماژول به **vcc**).

۳- کلید تغذیه ماژول را وصل کنید.

٤- در این حالت ماژول وارد مد **AT Command** می شود و بایستی LED وضعیت با فرکانس یک چشمک در ثانیه روشن و خاموش شود و LED اتصال خاموش باشد.

۵- در این حالت در محیط هایپر ترمینال دستور AT را وارد کرده و **Enter** کنید. اگر همه چیز بخوبی پیش رفته باشد بایستی OK را از سمت ماژول دریافت کنید. حتی اگر ERROR هم دریافت کنید باز هم نمایانگر ارتباط صحیح با ماژول است و شما در حال صحبت کردن با ماژول هستید! پس چند بار تلاش کنید تا بالاخره جواب OK را از ماژول بگیرید. (ERROR های پی در پی معمولا به خاطر سیم بندی شلوغ و خطاهای احتمالی ممکن است دریافت شود که در بار دوم یا سوم جای خود را به OK می دهد).

٦- دستور AT فقط برای تست ارتباط است و کاری انجام نمی دهد. دستورات بعدی را به ترتیب روی ماژول اجرا کنید و از هر کدام OK بگیرید. (توجه OK ها از طرف ماژول ارسال شده اند آنها را تایپ نکنید. بزرگ بودن و کوچک بودن حروف دستورات اهمیتی ندارد).

کدهای ATcommand بلوتوث در پیوست (ب) آمده است.

۷- و اما در توضیح دستورات بالا باید گفت که دستور اول یعنی AT فقط صحت ارتباط را چک می کند. دستور at+reset ماژول را ریست کرده و آماده تغییرات می شود. دستور at+orgl ماژول را به تنظیمات کارخانه بازمی گرداند (اگر قبلاً ماژول را تنظیم کرده باشید با این دستور تنظیمات قبلی پاک می شود و رمز عبور یا همان پین کد ماژول به ١٢٣٤ تغییر پیدا می کند). دستور at+name نام جاری ماژول را نمایش می دهد. دستور at+role نقش جاری ماژول را که بایستی ۰ باشد نمایش میدهد اگر عددی غیر از ۰ بود باید با دستور at+role=0 آن را به حالت ۰ که همان slave است تنظیم کنید.
دستور at+class مهمترین دستور این بخش می باشد که معمولا پاسخ ۰ را باز می گرداند. بایستی با دستور at+class=1 کلاس ماژول را به ۱ تغییر دهید. مجدداً با دستور at+class می توانید از ۱ شدن آن اطمینان حاصل کنید. با دستور at+name=BT- Module می توانید نام ماژول را به هر چه پس از علامت = نوشته شود تغییر دهید. این نام همان نامی است که موقع search کردن با موبایل می توانید آن را ببینید.

۸- تنظیمات تمام شده است. ماژول را خاموش کنید (قطع کلید تغذیه ماژول).

۹- قطع کلید حالت ماژول (قطع پایه ٣٤ از vcc).

۱۰- روشن کردن ماژول.

۱۱- در این حالت باید مشاهده کنیم که LED اتصال خاموش و LED وضعیت با سرعت بالا (۲ الی ۳ بار درثانیه) چشمک می زند. این نمایانگر مد دیتا بوده و در این حالت دیگر نمیتوان برای ماژول AT Command فرستاد. بلکه ماژول در حالت آماده برای جفت شدن می باشد.

۱۲- برنامه رایگان BlueTerm.apk برای اندروید جهت برقراری ارتباط.

۱۳- پس از اجرای برنامه درخواستی مبنی بر روشن کردن بلوتوث گوشی نمایش داده می شود، به در خواست جواب مثبت داده و در صورت تکرار درخواست بازهم پاسخ مثبت دهید. سپس پنجره تبلیغات را بسته و از منوی نرم افزار Connect device را لمس کرده و نام ماژول بلوتوث را از لیست دستگاه ها پیدا و انتخاب نمایید. توجه داشته باشید که در لیست دستگاه های این نرم افزار فقط دستگاه های Pair شده قرار می گیرند. به همین منظور بایستی قبل از اجرای این برنامه به بخش

بلوتوث گوشی یا تبلت خود رفته به صورت دستی ماژول را جستجو نمایید و آنرا با گوشی خود جفت کنید. پین کد پیشفرض ماژول نیز ۱۲۳٤ می باشد. پس از برقرار اتصال، پیغامی بر روی دستگاه اندرویدی نمایش داده می شود و LED اتصال بر روی ماژول روشن شده و LED وضعیت هر دوثانیه، جفت فلاش می زند. در این حالت هر متنی که در هایپر ترمینال ویندوز بنویسید در گوشی مشاهده می شود و هر چیزی که در ترمینال گوشی تایپ کنید نیز در هایپر ترمینال ویندوز تایپ می شود.

بدین ترتیب یک اتصال سریال بیسیم بین ماژول و موبایل یا تبلت برقرار می شود. از این به بعد کلید حالت ماژول همیشه بایستی قطع باشد و موقع روشن کردن ماژول LED وضعیت با سرعت چشمک می زند. تنظیمات نیز در حافظه ماژول باقی خواهد ماند. می توانید ماژول را از کامپیوتر جدا کرده و به پورت سریال یک میکروکنترلر که بر روی سرعت ۳۸٤۰۰-۸-N-1 تنظیم شده است متصل کنید و به تبادل اطلاعات بین میکرو و موبایل یا تبلت بپردازید. برنامه های اندرویدی اوپن سورس زیادی برای ارتباط با بلوتوث وجود دارد. می توانید یکی از آنها را از اینترنت دریافت کرده و با توجه به نیاز خود محیط گرافیکی برنامه را طراحی کنید. مثلا می توانید ترتیبی دهید که میکروکنترلر از طریق بلوتوث گوشی به اینترنت و یک وب سرور متصل شود و دمای محیط را بر روی شبکه اینترنت ارسال نماید و نیز بتوان دستگاهی را با استفاده از اینترنت روشن و خاموش نمود. (البته ماژول gprs، اترنت و wifi برای این منظور به صرفه تر است). می توانید به همین صورت دو ماژول را به صورت slave و master تنظیم کنید و بجای ماژول HMTR پروژه های میکروکنترلری خود را به یکدیگر متصل نمایید. برد ماژول بلوتوث بسیار کم بوده و در آزمایشی که ما انجام دادیم در محیط داخل ساختمان در حدود ۵ الی ٦ متر ارتباط به خوبی برقرار بود و به محض بیشتر شدن فاصله، لینک ارتباطی قطع می شد. البته در آزمایش دیگری که به صورت دید به دید انجام گرفت (از وسط راه پله آپارتمان سه طبقه از زیر زمین تا پشت بام) تا ۱۵ متر به خوبی جواب داد و با کوچکترین مانعی ارتباط قطع می شد. برد لینک ارتباطی به نوع گوشی موبایل یا تبلت نیز بستگی داشته و در کل استاندارد بلوتوث چیزی در حدود ۱۰ متر است. ارتباط بلوتوث برای شبکه های بیسیم رومیزی و کارهایی مثل آپدیت کردن برنامه یک میکرو (با استفاده از بوت لودر) فوق العاده عالی بوده و می توان به وسیله آن پروژه های پیشرفته ای را با میکرو کنترلر ها پیاده سازی نمود که کاملا بدون LCD و کی پد بوده و کنترل آنها از طریق تبلت یا موبایل صورت پذیرد.

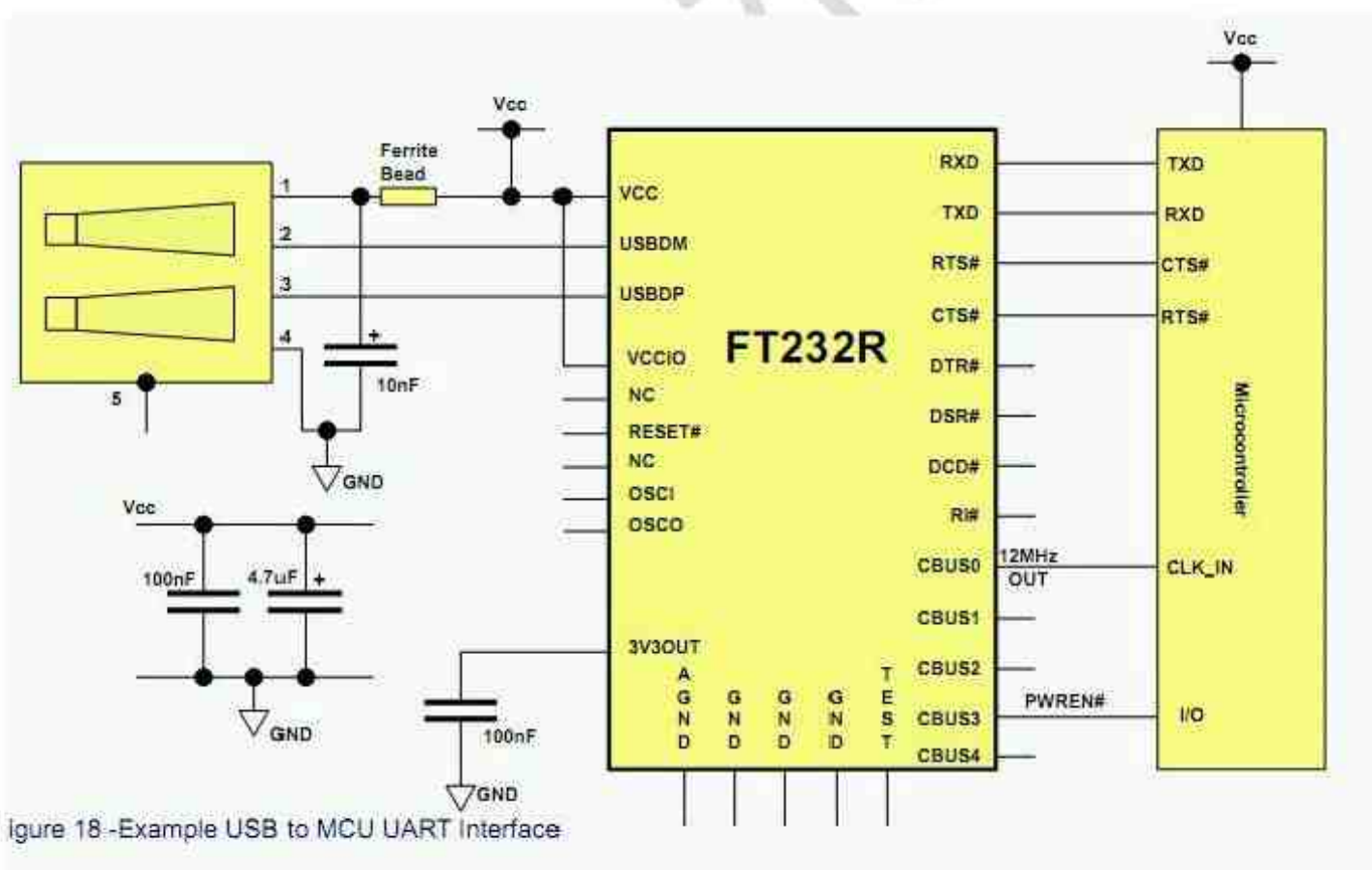
۲-۷-۱- معرفی آیسی مبدل USB به سریال FT232 :

تراشه FT323 ساخت ,FTDIChip یک مبدل USB به سریال است که برای برقراری ارتباط سریال آسنکرون بین کامپیوتر و تجهیزات خارجی از طریق درگاه USB طراحی شده است. با این تراشه می توان از ارتباط ,USB بدون وارد شدن به پروتکل پیچیده و مسائل مربوط به آن استفاده کرد. اهمیت این مسئله آن است که امروزه واسط USB به عنوان یک واسط استاندارد همه منظوره شناخته می شود و کاربرد واسط سریال RS232رو به کاهش است, به طوری که در نوت بوک ها امروزی, دیگر استفاده نمی شود. تراشه FT232 در چهار مدل مختلف FT232BL, FT232BM, FT232BQ و به تازگی FT232Rبه بازار عرضه شده است. عملکرد این تراشه ها, با وجود تفاوت های اندک, یکسان است. در این پروژه برای ارتباط میکروکنترلر AVR به کامپیوتر از تراشه FT232R استفاده شده است.

شکل (۲۰-۲) چیپ FT232RL ساخته شرکت FTDI

۲-۷-۲- نحوه کار با FT232RL:

تراشه FT232RL با ولتاژ 5V درگاه USB تغذیه می شود و نیازی به منبع تغذیه جداگانه نیست. در این پروژه، خروجی مدار برای جلوگیری از آسیب های احتمالی درگاه USB و مسائل مربوط به نویز می تواند توسط دو اپتوکوپلر (Optocoupler) محافظت شده است. در نهایت در خروجی، پایه های MTXD, MRXD, MGND و MVCC را خواهیم داشت که پایه های MTXD و MRXD مستقیما به میکروکنترلر ,پایه MGND به زمین مدار میکروکنترلر و MVCC به تغذیه مدار میکروکنترلر (تغذیه ۵ ولت) متصل می شوند. پس از ساخت این مدار، زمانی که برای اولین بار مدار را از طریق درگاه USB به کامپیوتر وصل کنید Windows ,آن را به عنوان یک سخت افزار جدید می شناسد و برنامه راه انداز (Driver) تراشه را درخواست می کند. با نصب برنامه راه انداز، یک درگاه سریال مجازی به فهرست درگاه های کامپیوتر اضافه خواهد شد. به این ترتیب در طرف کامپیوتر شما با یک درگاه سریال مجازی روبرو هستید و اطلاعات را از طریق این درگاه دریافت و ارسال کرد.



Igure 18 -Example USB to MCU UART Interface

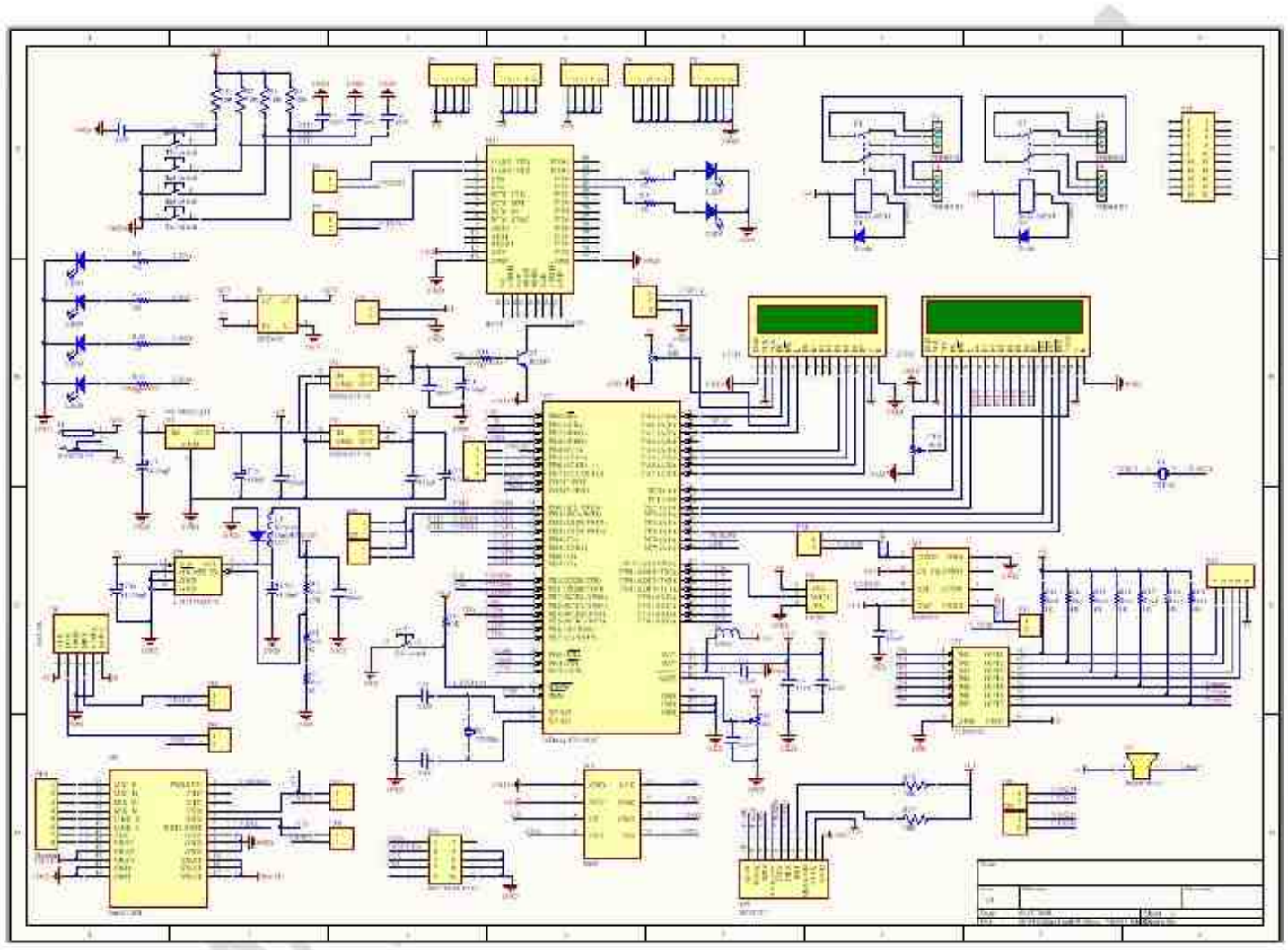
شکل (۲۱-۲) نحوه اتصال FT232R به میکروکنترلر

فصل سوم

شماتیک برد ارتباطی

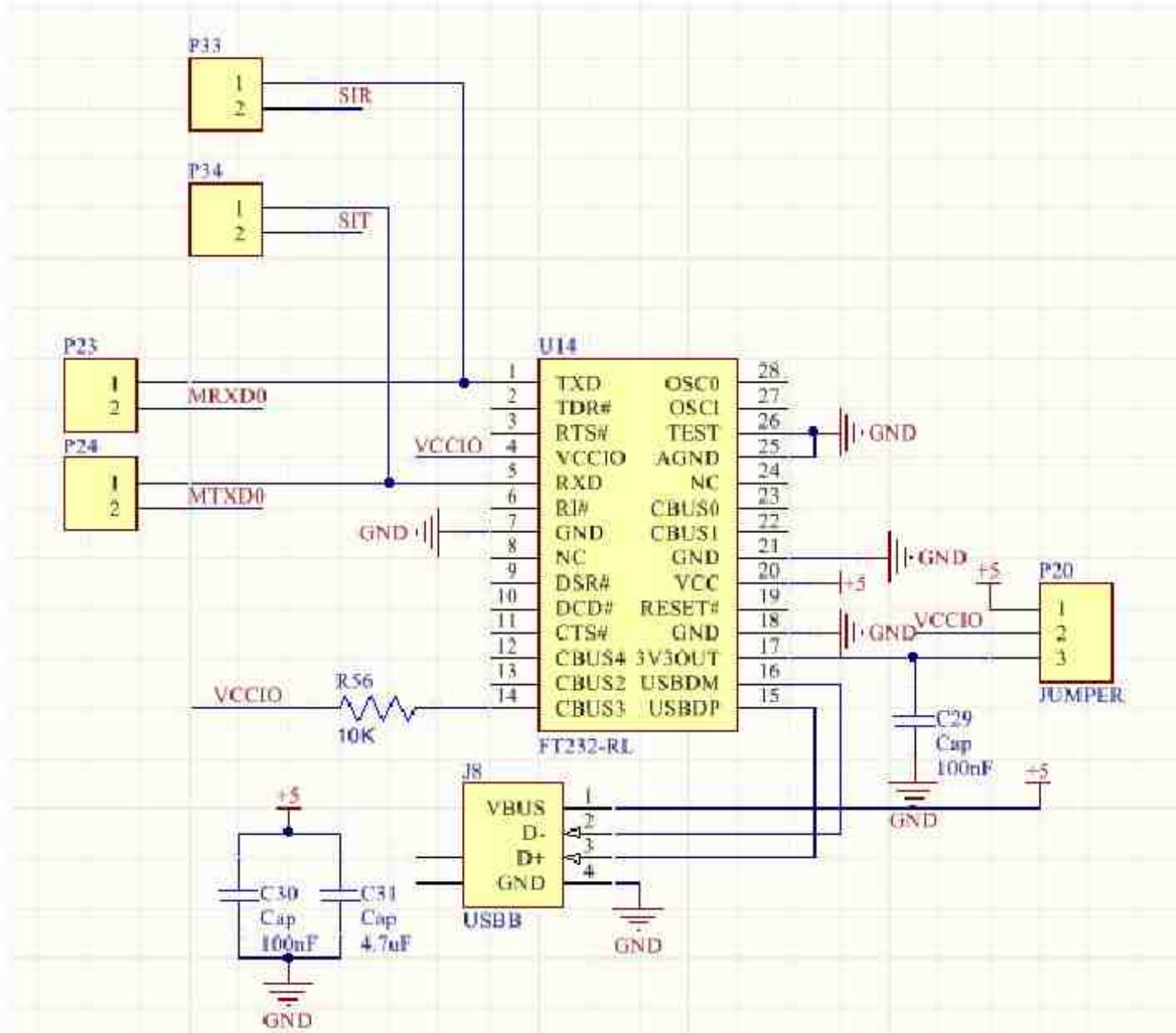
۳-۱- شماتیک های کشیده شده با نرم افزار Altium designer

۳-۱-۱- شماتیک مربوط به قسمت میکروکنترلر و اتصال آن به ماژول های ارتباطی بلوتوث ، وای فای ، رله ها ، LCD ها، سنسور دما ، تغذیه ها و رگولاتور ها می باشد :



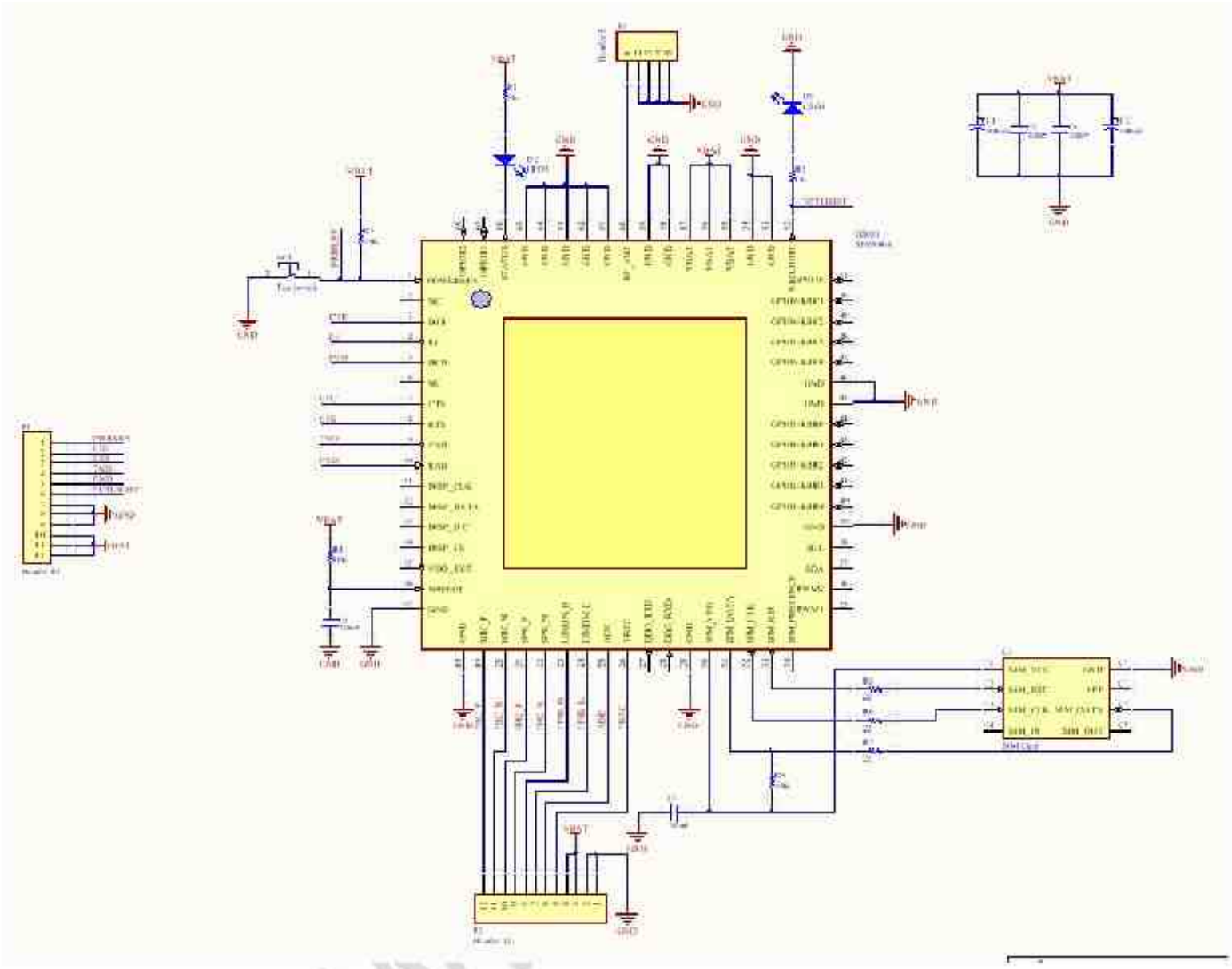
شکل (۱-۳) شماتیک برد مادر

۳-۱-۲ شماتیک مربوط به USB :



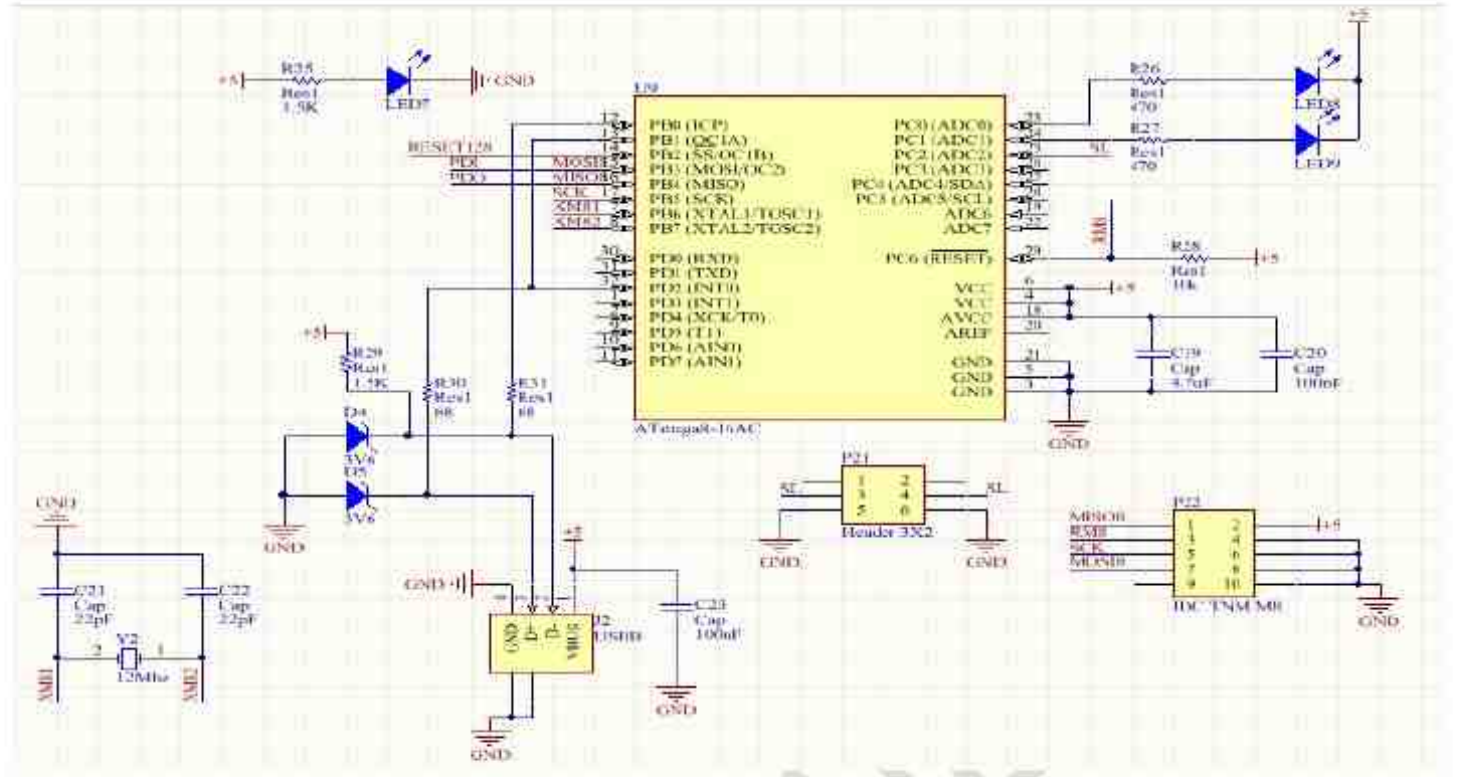
شکل (۲-۳) شماتیک بخش USB

۳-۱-۳- شماتیک زیر مربوط به ماژول SIM900.SMS می باشد:



شکل (۳-۳) شماتیک بخش SMS

۳-۱-٤- بر روی برد ساخته شده پروگرامر USBasp نیز طراحی شده است که برای پروگرام کردن میکروی اصلی روی برد دیگر نیازی به اتصال پروگرامر از خارج برد نباشد.



شکل (٤-۳) شماتیک بخش پروگرامر آنبرد USBasp

پیوست ها

پیوست (الف) : دستورات ATcommand مربوط به SIM900

- دستورات ماژول SIM900

ماژول SIM900 به صورت کاراکتری دستورات را دریافت می کند و به همان صورت هم به آنها پاسخ می دهد. در انتهای هر دستور باید دو کاراکتر CR و LF که معادل عددی آنها به ترتیب ۱۰ و ۱۳ است ارسال شوند. این دو کاراکتر در جدول ASCII که پیش از این در رابطه با آن بحث شده است تعریف شده اند. به این شکل ارتباطی بین یک ماژول و CPU پروتکل ATcommand می گویند. ماژول پس از دریافت هر دستور پاسخ آن را ارسال می کند.

- دستور AT

این دستور برای تست سلامت ارتباط با ماژول می باشد. ماژول در پاسخ به این دستور OK پاسخ می دهد.

AT

OK

- دستور ATI

ماژول در پاسخ به این دستور مدل و ورژن خود را ارسال می کند.

ATI

SIM900 R11.0

OK

- دستور **AT+CSQ**

ماژول در پاسخ به این دستور دو عدد را باز می گرداند. عدد اول قدرت سیگنال آنتن را با عددی بین ۰ تا ۳۲ برمی گرداند و عدد دوم نشاندهنده جزئیاتی است که در اینجا برای ما اهمیتی ندارد.

AT+CSQ

+CSQ: 14,0

OK

- دستور **AT+CBC**

ماژول در پاسخ به این دستور سه عدد را برمی گرداند. عدد اول نشان دهنده در حال شارژ بودن باطری است. عدد دوم میزان شارژ باطری را به صورت عددی بین ۰ تا ۱۰۰ نشان می دهد و عدد سوم ولتاژ باطری را بر حسب میلی ولت نشان می دهد.

AT+CBC

+CBC: 0,100,4338

OK

- دستور **AT+CMGF (Select sms message format)**

این دستور برای انتخاب فرمت نوشتن و خواندن پیامک در SIM900 است. این ماژول با دو فرمت Text و PDU می تواند پیامک ها را سرویس دهی کند. هر چند فرمت PDU کاملتر است ولی بسیار پیچیده بوده و در اینجا به دلیل سادگی فقط با فرمت Text کار می کنیم. برای انتخاب فرمت Text کافی است دستور AT+CMGF=1 را به ماژول بفرستید. ماژول در پاسخ OK بر می گرداند.

AT+CMGF=1

OK

- دستور **AT+CMGS (Send SMS)**

این دستور در فرمت Text و PDU رفتار متفاوتی دارد که در اینجا فرمت Text آن را تشریح می کنیم. ابتدا باید شمار گیرنده را در انتهای دستور وارد شود:

AT+CMGS="09191368415"

در پاسخ این دستور ماژول کاراکتر '>' را بر می گرداند که به این معنی است که ماژول آماده دریافت متن پیامک است :

AT+CMGS="09191368415"

> Hello SIM900 . i am student

پس از وارد کردن متن پیام بایستی کاراکتر 0x1a را ارسال کنیم. ماژول با دریافت این کد شروع به ارسال پیامک می کند و در صورت عدم موفقیت کلمه ERROR را برمی گرداند و در صورت ارسال موفق OK پاسخ می دهد :

AT+CMGS="09191368415"

<Hello SIM900 . i am student.

+CMGS: 14 OK

- دستور **AT+CMGL (List SMS messages)**

این دستور برای خواندن پیامک های موجود در سیم کارت است و به چند شکل می توان از آن استفاده کرد :

جدول ۱-الف. ATcommand های SIM900

عملکرد	دستور
خواندن تمام پیامک های دریافتی خوانده نشده.	AT+CMGL="REC UNREAD"

خواندن تمام پیامک های دریافتی خوانده شده.	AT+CMGL="REC READ"
خواندن تمام پیامک های ذخیره شده اما ارسال نشده.	AT+CMGL="STO UNSENT"
خواندن تمام پیامک های ذخیره شده و ارسال شده.	AT+CMGL="STO SENT"
خواندن تمام پیامک ها بدون قید و شرط.	AT+CMGL="ALL"

ماژول در پاسخ به این دستور متن تمام پیامک ها را با زمان دریافت و سایر مشخصاتشان بر میگرداند:

AT+CMGL="REC UNREAD"

+CMGL: 2,"REC UNREAD","+989191368415","E57A@","11/09/28,19:30:01+14"

Hello

This is Arash Fattahi

OK

- دستور **AT+CMGD (Delete SMS messages)**

این دستور برای حذف پیامک های موجود در سیم کارت است و به دنبال آن دو عدد باید ارسال کنیم.

عدد اول را **index** می نامیم و عدد دوم را **delflag** که این دو عدد عملکرد دستور را کنترل می کنند. در جدول زیر نحوه عملکرد دستور را بر اساس مقادیر مختلف **delflag** مشاهده می کنید.

جدول ۲- الف. دستورات خواندن و نوشتن پیام در SIM900

عملکرد	Delflag
حافظه :**index** پاک کردن پیامک از خانه شماره به عنوان مثال دستور زیر پیامک موجود در خانه شماره ۲۷ حافظه سیم کارت را پاک می کند AT+CMGD=27,0 OK	0
حذف تمام پیامک های دریافتی خوانده شده از حافظه. AT+CMGD=0,1	1

OK	
حذف تمام پیامک های دریافتی خوانده شده و ذخیره شده ارسال شده. AT+CMGD=0,2 OK	2
حذف تمام پیامک ها غیر از پیامهای دریافتی خوانده نشده. AT+CMGD=0,3 OK	3
حذف تمام پیامک ها بدون قید و شرط. AT+CMGD=0,4 OK	4

پیوست (ب) : دستورات **ATcommand** مربوط به **BlueTooth**

AT

OK

at+reset

OK

AT+ORGL

OK

at+name

+NAME:H-C-2010-06-01

OK

at+role

+ROLE:0

OK

at+class

+CLASS:0

OK

at+class=1

OK

at+class

+CLASS:1

OK

at+name=BT-Module

OK

at+name

+NAME:BT-Module

پیوست (ج) : کد برنامه نوشته شده بر روی میکرو

```
#include <mega128.h>

#include <string.h>

#include <delay.h>


#include <stdlib.h>

// Alphanumeric LCD Module functions

#include <alcd.h>


// Specify that a new putchar function will be used instead of the one from stdio.h

#define _ALTERNATE_PUTCHAR_

#include <stdio.h>

void clr(void);

//unsigned char flag=0,b;

unsigned char i=0,j,k,led,b,mu=0;

char c,s[6],tr[10],vr[10],t[]="TEMP=",v[]="V=",u[]="REZA FATTAHI";

unsigned int a;


unsigned char flag=0,flr=0,flc=0,r;

char d[200],*wf,*sms;

#ifndef RXB8

#define RXB8 1

#endif
```

```
#ifndef TXB8

#define TXB8 0

#endif


#ifndef UPE

#define UPE 2

#endif


#ifndef DOR

#define DOR 3

#endif


#ifndef FE

#define FE 4

#endif


#ifndef UDRE

#define UDRE 5

#endif


#ifndef RXC

#define RXC 7

#endif
```

```
#define FRAMING_ERROR (1<<FE)

#define PARITY_ERROR (1<<UPE)

#define DATA_OVERRUN (1<<DOR)

#define DATA_REGISTER_EMPTY (1<<UDRE)

#define RX_COMPLETE (1<<RXC)


// USART1 Receiver buffer

// This flag is set on USART1 Receiver buffer overflow

bit rx_buffer_overflow1;


// USART1 Receiver interrupt service routine

interrupt [USART1_RXC] void usart1_rx_isr(void)

{

  char data;


 if(mu==1) goto smu1;

 if(mu==2) goto smu2;
```

```
if(mu==3) goto smu3;


smu1:
if(flag==1) goto ss;
data=UDR1;
d[i]=data;
lcd_putchar(d[i]);
i++;
//lcd_putchar(data);
if(data>30) lcd_putchar(data);
if(data=='#') flr=1;
goto ss;
//---------------------------------
smu2:



if(flag==1) goto ss;
data=UDR1;
if(data==':'){flc=1;goto ss;}
 if(flc==1)
  {
   d[i]=data;
  //if(data>30) lcd_putchar(d[i]);
   i++;
```

```
   flc=0;
   }
   if(data=='#') flr=1;
//if(data>30&&data<='z') lcd_putchar(data);
//if(data==10) {flr=1;}
goto ss;
//---------------------------------------
smu3:


//if(flag==1) goto ss;
data=UDR1;
d[i]=data;
//lcd_putchar(d[i]);
i++;
 if(data>=30&&data<='z'&&flag==1) lcd_putchar(data);


//-------flag 2 ---------
if(flag==2)
{  flag=3;
   //lcd_gotoxy(0,1);lcd_putsf("flag=");lcd_putchar(flag+48);
}
  ss:
 }
```

```
/*
unsigned char rx_counter1;
unsigned char rx_buffer1;
// Get a character from the USART1 Receiver buffer
#pragma used+
char getchar1(void)
{
char data;
while (rx_counter1==0);
data=rx_buffer1[rx_rd_index1++];
#if RX_BUFFER_SIZE1 != 256
if (rx_rd_index1 == RX_BUFFER_SIZE1) rx_rd_index1=0;
#endif
#asm("cli")
--rx_counter1;
#asm("sei")
return data;
}
#pragma used-
// Write a character to the USART1 Transmitter
#pragma used+
*/
```

```
void putchar1(char c)
{
while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
UDR1=c;
}
#pragma used-
```

```
// Standard Input/Output functions



// Specify the output types
#define USART0 0
#define USART1 1
#define LCD 2

// This variable will specify to which peripheral the output of putchar will be directed
unsigned char poutput;
```

```
// Define the new putchar function
void putchar(char c)
{
switch (poutput)
      {
      case USART0: // the output will be directed to USART0
          while ((UCSR0A & DATA_REGISTER_EMPTY)==0);
          UDR0=c;
      break;

      case USART1: // the output will be directed to USART1
          while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
          UDR1=c;
      break;

      case LCD:    // the output will be directed to the LCD
          lcd_putchar(c);
      };
}

// Declare your global variables here
```

```
void main(void)
{
poutput=USART1;
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
```

```
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In

// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T

PORTC=0x00;

DDRC=0x00;


// Port D initialization

// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In

// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T

PORTD=0x00;

DDRD=0x00;


// Port E initialization

// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In

// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T

PORTE=0x00;

DDRE=0xf0;


// Port F initialization

// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
```

```
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T

PORTF=0x00;

DDRF=0xf0;


// Port G initialization

// Func4=In Func3=In Func2=In Func1=In Func0=In

// State4=T State3=T State2=T State1=T State0=T

PORTG=0x00;

DDRG=0x00;


// Timer/Counter 0 initialization

// Clock source: System Clock

// Clock value: Timer 0 Stopped

// Mode: Normal top=0xFF

// OC0 output: Disconnected

ASSR=0x00;

TCCR0=0x00;

TCNT0=0x00;

OCR0=0x00;


// Timer/Counter 1 initialization

// Clock source: System Clock

// Clock value: Timer1 Stopped
```

```
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// OC1C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;
```

```
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer3 Stopped
// Mode: Normal top=0xFFFF
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
```

```
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;
```

```
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

ETIMSK=0x00;

// USART0 initialization
// USART0 disabled
UCSR0B=0x00;

// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: On
// USART1 Mode: Asynchronous
// USART1 Baud Rate: 9600
UCSR1A=0x00;
UCSR1B=0x98;
UCSR1C=0x06;
UBRR1H=0x00;
UBRR1L=0x4D;

// Analog Comparator initialization
// Analog Comparator: Off
```

```
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

// Alphanumeric LCD initialization
// Connections specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTA Bit 0
// RD - PORTA Bit 1
// EN - PORTA Bit 2
// D4 - PORTA Bit 4
// D5 - PORTA Bit 5
```

```
// D6 - PORTA Bit 6
// D7 - PORTA Bit 7
// Characters/line: 12
lcd_init(16);

// Global enable interrupts
#asm("sei")
  lcd_clear();

        lcd_putsf("SEL COM TERMINAL");
      i=0;
      flag=2;
      while (1)
     {

      if(PIND.0==0)
      {
       delay_ms(20);
       mu++;
       if(mu==4) mu=1;
      if(mu==1) {lcd_clear();lcd_gotoxy(0,1);lcd_putsf("
BLUETOOTH");delay_ms(1000);lcd_gotoxy(0,0);i=0;j=0;flr=0;}
      if(mu==2) {lcd_clear();lcd_gotoxy(0,1);lcd_putsf("
WIFI");delay_ms(1000);lcd_gotoxy(0,0);i=0;j=0;flr=0;flag=1;}
```

```
      if(mu==3) {lcd_clear();lcd_gotoxy(0,1);lcd_putsf("
SMS");delay_ms(1000);lcd_gotoxy(0,0);i=0;j=0;flr=0;flag=1;}
      while(PIND.0==0);
      delay_ms(50);
      }


     if(mu==1)
     {

     if(flr==1)
     {
     #asm("cli")
     j=0;
     lcd_gotoxy(0,1);
     while(d[j]!='#')
      //while(j<9)
     {
      lcd_putchar(d[j]);
      //wf[j]=d[j];
      j++;
      delay_ms(100);
     }
     //lcd_putsf("REZA");while(1);
```

```
//goto end;



/*wf=strstrf(d,"$$");
lcd_clear();
j=2;
while(wf[j]!=13)
{
 lcd_putchar(wf[j]);
 j++;
}  */
/*  if(wf[2]=='l'|wf[2]=='L') goto s1;
 if(wf[2]=='r'|wf[2]=='R') goto s2;
 if(wf[2]=='b'|wf[2]=='B') goto s3;  */


 if(d[2]=='l'|d[2]=='L') goto s1;
 if(d[2]=='r'|d[2]=='R') goto s2;
 if(d[2]=='b'|d[2]=='B') goto s3;


 ////------------------------LED-------------
 s1:
 //lcd_putsf("REZA");while(1);
   if(d[6]=='n'|d[6]=='N')
```

```
{
 led=d[7]-48;
 if(led==4) PORTE.4=1;
 if(led==5) PORTE.5=1;
 if(led==6) PORTE.6=1;
 if(led==7) PORTE.7=1;
 }

 if(d[6]=='f'|d[6]=='F')
{
 led=d[8]-48;
 if(led==4) PORTE.4=0;
 if(led==5) PORTE.5=0;
 if(led==6) PORTE.6=0;
 if(led==7) PORTE.7=0;
 }
 goto end;
 ////------------------------RELAY-------------
 s2:
  if(d[6]=='n'|d[6]=='N')
 {
 r=d[7]-48;
 if(r==1) PORTF|=1<<5;
 if(r==2) PORTF|=1<<6;
```

```
}
 if(d[6]=='f'|d[6]=='N')
{
 r=wf[8]-48;
 if(r==1) PORTF&=0b11011111;
 if(r==2) PORTF&=0b10111111;
}

goto end;


////------------------------BUZZER-------------
s3:
 if(d[6]=='n'|d[6]=='N')
{
PORTF|=1<<4;
}
 if(d[6]=='f'|d[6]=='N')
{
PORTF&=0b11101111;
}
 if(d[6]=='e'|d[6]=='E')
{
PORTF|=1<<4;
```

```
delay_ms(30);
PORTF&=0b11101111;
PORTF|=1<<4;
delay_ms(30);
PORTF&=0b11101111;
}
 goto end;



end:
lcd_gotoxy(0,0);
lcd_putsf("              ");
        lcd_gotoxy(0,1);
        lcd_putsf("   BLUETOOTH");
        flr=0;
        for(i=0;i<100;i++) {d[i]=0;}
        i=0;
        lcd_gotoxy(0,0);
        #asm("sei")
    }
    }
    //-------------------------------------------------MU=2-----------------------------
----
    if(mu==2)
```

```
{
if(flag==1)
{
#asm("sei")
putchar1('A');
delay_ms(1);
putchar1('T');
delay_ms(1);
putchar1('+');
delay_ms(1);
putchar1('C');
delay_ms(1);
putchar1('W');
delay_ms(1);
putchar1('M');
delay_ms(1);
putchar1('O');
delay_ms(1);
putchar1('D');
delay_ms(1);
putchar1('E');
delay_ms(1);
putchar1('=');
delay_ms(1);
```

```
putchar1('3');
delay_ms(1);
putchar1(13);
delay_ms(1);
putchar1(10);
delay_ms(1);
//flag=2;
delay_ms(1);
lcd_clear();
////////////////////////////////////AT+CIFSR
putchar1('A');
delay_ms(1);
putchar1('T');
delay_ms(1);
putchar1('+');
delay_ms(1);
putchar1('C');
delay_ms(1);
putchar1('I');
delay_ms(1);
putchar1('F');
delay_ms(1);
putchar1('S');
delay_ms(1);
```

```
putchar1('R');
delay_ms(1);
putchar1(13);
delay_ms(1);
putchar1(10);
delay_ms(1);
//flag=2;
delay_ms(1);
lcd_clear();
//////////////////AT+CIPMUX=1
flag=1;
 putchar1('A');
delay_ms(1);
putchar1('T');
delay_ms(1);
putchar1('+');
delay_ms(1);
putchar1('C');
delay_ms(1);
putchar1('I');
delay_ms(1);
putchar1('P');
delay_ms(1);
putchar1('M');
```

```
delay_ms(1);
putchar1('U');
delay_ms(1);
 putchar1('X');
delay_ms(1);
 putchar1('=');
delay_ms(1);
 putchar1('1');
delay_ms(1);
putchar1(13);
delay_ms(1);
putchar1(10);
delay_ms(1);
//flag=2;
 delay_ms(1);
lcd_clear();
//////////////////AT+CIPSERVER=1,1344
  putchar1('A');
delay_ms(1);
putchar1('T');
delay_ms(1);
putchar1('+');
delay_ms(1);
putchar1('C');
```

```
delay_ms(1);
putchar1('I');
delay_ms(1);
putchar1('P');
delay_ms(1);
putchar1('S');
delay_ms(1);
putchar1('E');
delay_ms(1);
 putchar1('R');
delay_ms(1);
 putchar1('V');
delay_ms(1);
putchar1('E');
delay_ms(1);
 putchar1('R');
delay_ms(1);
 putchar1('=');
delay_ms(1);
 putchar1('1');
  putchar1(',');
delay_ms(1);
 putchar1('1');
delay_ms(1);
```

```
putchar1('3');
delay_ms(1);
 putchar1('4');
delay_ms(1);
 putchar1('4');
delay_ms(1);
putchar1(13);
delay_ms(1);
putchar1(10);
delay_ms(1);


 delay_ms(1);
lcd_clear();
//////////////////AT+CWSAP="FATTAHI","1234",3,0
  putchar1('A');
delay_ms(1);
putchar1('T');
delay_ms(1);
putchar1('+');
delay_ms(1);
putchar1('C');
delay_ms(1);
putchar1('W');
delay_ms(1);
```

```
putchar1('S');
delay_ms(1);
putchar1('A');
delay_ms(1);
putchar1('P');
delay_ms(1);
 putchar1('=');
delay_ms(1);
 putchar1('"');
delay_ms(1);
putchar1('F');
delay_ms(1);
 putchar1('A');
delay_ms(1);
 putchar1('T');
delay_ms(1);
 putchar1('T');
  putchar1('A');
delay_ms(1);
 putchar1('H');
delay_ms(1);
 putchar1('I');
 delay_ms(1);
  putchar1('"');
```

```
delay_ms(1);
  putchar1(',');
delay_ms(1);
 putchar1('"');
delay_ms(1);
 putchar1('1');
delay_ms(1);
putchar1('2');
delay_ms(1);
 putchar1('3');
delay_ms(1);
 putchar1('4');
delay_ms(1);
 putchar1('"');
delay_ms(1);
 putchar1(',');
delay_ms(1);
 putchar1('3');
delay_ms(1);
 putchar1(',');
delay_ms(1);
 putchar1('0');
delay_ms(1);
```

```
putchar1(13);
delay_ms(1);
putchar1(10);
delay_ms(1);

delay_ms(1);
//lcd_clear();
 i=0;
 flag=2;
 flr=0;
 j=0;
}
lcd_gotoxy(0,1);lcd_putsf("   WIFI READY");

//while(1);
if(flr==1)
{
//lcd_gotoxy(0,1);lcd_putsf("   WIFI READY");
//lcd_putsf("reza");while(1);
 #asm("cli")
 wf=strstrf(d,"$$");
 lcd_clear();
```

```
j=2;
lcd_gotoxy(0,0);
//while(wf[j]!=13)
while(j<12)
{
 lcd_putchar(wf[j]);
 j++;
}
//while(1);
if(wf[2]=='l'|wf[2]=='L') goto s21;
if(wf[2]=='r'|wf[2]=='R') goto s22;
if(wf[2]=='b'|wf[2]=='B') goto s23;

////------------------------LED-------------
s21:
  if(wf[6]=='n'|wf[6]=='N')
  {
   led=wf[7]-48;
   if(led==4) PORTE.4=1;
   if(led==5) PORTE.5=1;
   if(led==6) PORTE.6=1;
   if(led==7) PORTE.7=1;
   }
```

```
if(wf[6]=='f'|wf[6]=='F')
{
led=wf[8]-48;
if(led==4) PORTE.4=0;
if(led==5) PORTE.5=0;
if(led==6) PORTE.6=0;
if(led==7) PORTE.7=0;
}
goto end2;
////------------------------RELAY-------------
s22:
 if(wf[6]=='n'|wf[6]=='N')
{
r=wf[7]-48;
if(r==1) PORTF|=1<<5;
if(r==2) PORTF|=1<<6;
}
 if(wf[6]=='f'|wf[6]=='N')
{
 r=wf[8]-48;
 if(r==1) PORTF&=0b11011111;
 if(r==2) PORTF&=0b10111111;
}
```

```
goto end2;



////-------------------------BUZZER--------------
s23:
 if(wf[6]=='n'|wf[6]=='N')
{
PORTF|=1<<4;
}
 if(wf[6]=='f'|wf[6]=='N')
{
PORTF&=0b11101111;
}
 if(wf[6]=='e'|wf[6]=='E')
{
PORTF|=1<<4;
delay_ms(30);
PORTF&=0b11101111;
PORTF|=1<<4;
delay_ms(30);
PORTF&=0b11101111;
}
 goto end2;
```

```
        end2:
        flr=0;
        i=0;
        #asm("sei")
    }

}

//////////////////////----------------------------MU=3-----------------------------
--
    if(mu==3)
    {
    //---------- init SIM900 -------
    if(flag==1)
    {
clr();
flag=1;
delay_ms(500);
printf("AT%c%c",13,10);
delay_ms(500);
printf("AT+CMGF=1%c%c",13,10);
delay_ms(1000);
itoa(i,s);
```

```
lcd_puts(s);
delay_ms(1000);
lcd_clear();

_lcd_write_data(0xf);
printf("AT+CMGD=0,4%c%c",13,10);
delay_ms(500);
lcd_clear();

_lcd_write_data(0xf);
lcd_putsf("Ready:");
clr();
flag=2;
j=0;
}
//---------- ready for recive sms -------

//flag=2;
//j=0;


    /*
     if(PIND.5==0)
    {
```

```
lcd_clear();
lcd_puts(u);//"REZA FATTAHI"
delay_ms(200);
printf("AT+CMGS=\"09123083217\"%c%c",13,10);
delay_ms(500);
printf("%s%c%c",u,13,10);
delay_ms(50);
printf("%c",26);
while(PIND.5==0);
delay_ms(200);
} */
//----------------------- read sms flag=3 ------
if(flag==3)
{
delay_ms(500);
printf("AT+CMGL=\"REC UNREAD\"%c%c",13,10);
delay_ms(500);

lcd_clear();
_lcd_write_data(0xf);
//----------------------LED ON OR OFF----------------
sms=strstrf(d,"$$");
lcd_gotoxy(0,0);
for(j=2;j<9;j++)
```

```
{
if(sms[j]>=30&&sms[j]<='z')
lcd_putchar(sms[j]);
}
if(sms[2]=='l') goto s31;
if(sms[2]=='r') goto s32;
if(sms[2]=='b') goto s33;

s31:

//---------- ON LED -------
if((sms[5]=='o'||sms[5]=='O')&&(sms[6]=='n'||sms[6]=='N'))
{
led=sms[7]-48;
if(led==0) PORTC.0=1;
if(led==1) PORTC.1=1;
if(led==2) PORTC.2=1;
if(led==3) PORTC.3=1;
if(led==4) PORTE.4=1;
if(led==5) PORTE.5=1;
if(led==6) PORTE.6=1;
if(led==7) PORTE.7=1;
}
```

```
    //---------- OFF LED ----------

if((sms[5]=='o'||sms[5]=='O')&&(sms[6]=='f'||sms[6]=='F')&&(sms[7]=='f'||
|sms[7]=='F'))
    {
    led=sms[8]-48;
    if(led==0) PORTC.0=0;
    if(led==1) PORTC.1=0;
    if(led==2) PORTC.2=0;
    if(led==3) PORTC.3=0;
    if(led==4) PORTE.4=0;
    if(led==5) PORTE.5=0;
    if(led==6) PORTE.6=0;
if(led==7) PORTE.7=0;
}
goto s34;

////------------------------RELAY-------------
    s32:
     if(sms[6]=='n'|sms[6]=='N')
    {
    r=sms[7]-48;
    if(r==1) PORTF|=1<<5;
    if(r==2) PORTF|=1<<6;
```

```
    }
     if(sms[6]=='f'|sms[6]=='F')
    {
     r=sms[8]-48;
     if(r==1) PORTF&=0b11011111;
     if(r==2) PORTF&=0b10111111;
    }
goto s34;

 ////------------------------BUZZER-------------
    s33:
     if(sms[6]=='n'|sms[6]=='N')
{
PORTF|=1<<4;
}
 if(sms[6]=='f'|sms[6]=='F')
{
PORTF&=0b11101111;
}
 if(sms[6]=='e'|sms[6]=='E')
{
PORTF|=1<<4;
delay_ms(30);
PORTF&=0b11101111;
```

```
        PORTF|=1<<4;
        delay_ms(30);
        PORTF&=0b11101111;
        }
    //-----------------------------
    s34:
    delay_ms(1000);
    printf("AT+CMGD=0,4%c%c",13,10);
    delay_ms(1000);
    clr();
    flag=1;
}
    /*
//--------------------- Read&Write TEMP & SPEED ---------------
    a=read_adc(0);
    b=a/4;
    itoa(b,tr);
    lcd_gotoxy(0,1);
    lcd_putsf("TMP=");
    lcd_puts(tr);
    lcd_putsf("   ");
    a=read_adc(1);
    b=a/4;
    itoa(b,vr);
```

```
    lcd_gotoxy(8,1);
    lcd_putsf("SPD=");
    lcd_puts(vr);
    lcd_putsf("    ");
    delay_ms(500);
    */



    }
    }
}

//----------------FUNCTION-------------------


void clr(void)
{
i=0;
j=0;
for(k=0;k<200;k++)
d[k]=255;
flag=1;
}
```

فهرست منابع

منابع

۱-دیتاشیت ATmega128 شرکت Atmel .

۲-دیتاشیت ماژول ESP8266

۳-دیتاشیت HC-05 Bluetooth Module

Abstract:

A wireless communication terminal, which forms a wireless network with a plurality of other wireless communication terminals, includes a detection portion to detect a first communication management signal periodically transmitted via broadcast from another wireless communication terminal prior to communication of a data signal, a communication control portion to determine whether to transmit a second communication management signal to the wireless communication terminal to transmit the first communication management signal based on a detection status of the first communication management signal, and a wireless communication portion serving as a transmission portion to transmit the second communication management signal via unicast to the wireless communication terminal as a transmission source of the first communication management signal and as a reception portion to receive an acknowledge signal transmitted via unicast in response to the second communication management signal.

Keywords: Wireless Communication,ATmega128 microcontroller, Serial wirelessComunicated Modules.

Faculty of Technical and Engineering, Department of Electrical

"B.Sc" Thesis

on Electronic

Title:

USB , Wifi , SMS and Bluetooth Commiunication Board

consulting Advisor:

Dr. Seyyed Hossein Pishgar

By:
Arash Fattahi

Winter 2016