سیستم قرائت خودکار کنتور با استفاده از شبکه مشZigBee

Melec.ir

اردیبهشت ۱۳۹۰

.

فهرست مطالب

⇐ مواردي كه در گزارش به صورت Hyperlink مي‌باشد، به فايل-هاي مربوطه در CD گزارش ارجاع داده شده است.

تکنولوژي بي‌سيم ZigBee با استفاده از تراشه XBee PRO Series 2

۱-۱- استاندارد IEEE 802.15.4 و Zigbee

استاندارد IEEE 802.15.4 پروتکل و ارتباط بین وسایل را از طریق مخابرات رادیویی بی سیم در یک شبکه شخصی PAN تعریف می کند. این استاندارد در باند فرکانسی ISM (صنعتی، علمی و پزشکی)، ۸۶۸ MHz در اروپا، ۹۱۵ MHz در آمریکا و ۲/٤ GHz جهانی کار می کند. هدف ایجاد استانداردی با پیچیدگی بسیار پایین، هزینه بسیار پایین، مصرف توان بسیار پایین و نرخ داده پایین برای ارتباط بی سیم بین وسایل ارزان است. استاندارد از روش دستیابی CDMA/CA استفاده می کند و توپولوژی های مش و ستاره را پشتیبانی می کند. حداکثر نرخ داده خام ۲۵۰kbps است که اکثر نیازمندی های کنترل و اتوماسیون صنعتی متعدد را برای ارتباط بی سیم برآورده می سازد.

در حالیکه استاندارد IEEE 802.15.4 لایه فیزیکی و پروتکلهای MAC را مشخص می کند، پروتکلهای لایه های بالاتر توسط این استاندارد پوشیده نمی شود. اتحادیه ZigBee سازمانی از شرکتهای صنعتی است که تکنولوژی های مبتنی بر استاندارد IEEE 802.15.4 را ترفیع می دهد و مجموعه مشخصات پروتکلهای سطح بالای مبتنی بر استاندارد IEEE 802.15.4 را منتشر می سازد. تکنولوژی ZigBee می تواند در کاربردهای مختلف صنعتی، ساختمانی، پزشکی، تجاری و غیره استفاده شود.

تکنولوژي ZigBee توپولوژي‌هاي شبکه مختلفي دارد که شامل ستاره (Star)، خوشه (Cluster Tree) و مش (Mesh) مي‌باشد. اين توپولوژي‌هاي مختلف در شکل۱-۱ نشان داده شده‌اند. سه نوع Device در هر شبکه مش ZigBee وجود دارد که عبارتند از:

Coordinator: در هر شبکه تنها يك Coordinator وجود دارد که وظيفه ذخيره‌سازي اطلاعات مربوط به شبکه را بر عهده داشته و همچنين مسير بهينه انتقال اطلاعات بين دو نود را تعيين مي‌کند.

Router: يك واسط براي انتقال اطلاعات بين نودها مي‌باشد.

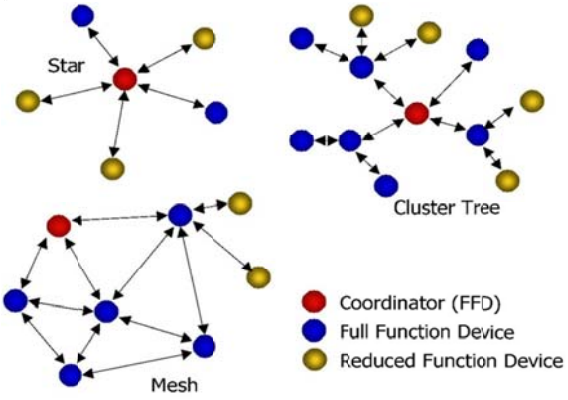
End Device: کمترين عملکرد را در شبکه دارد و تنها مي‌تواند با يك Router و يا با Coordinator ارتباط برقرار کند و قادر نيست اطلاعات را بين نودهاي همسايه انتقال دهد.

ويژگي اصلي شبکه مش که آن را از ساير توپولوژي‌ها متمايز مي-سازد اين است که در اين توپولوژي علاوه بر ارسال و دريافت اطلاعات بي‌سيم توسط هر نود، Router مي‌تواند به عنوان يك واسط براي انتقال اطلاعات بين نودهاي همسايه به کار رود. بدين ترتيب در شبکه مش بي سيم يك پکت داده مسير خود را به سمت مقصد نهايي پيدا کرده، از نودهاي مياني با لينکهاي ارتباطي مطمئن عبور مي کند. ساير ويژگي‌هاي مطلوب شبکه مش ZigBee شامل موارد زير است:

۱- اگر يك نود به هر دليلي (نظير ورود تداخل RF قوي) دچار مشکل شود، پيام به طور خودکار از طريق مسيرهاي ديگر به سمت مقصد هدايت مي شود.

۲- فاصله بين نودهاي بي سيم مي تواند کوتاه شود. اينکار کيفيت لينك بين نودها را به ميزان قابل توجهي افزايش مي دهد، بدون اينکه توان ارسالي هر نود افزايش يابد.

۳-به سادگي با اضافه كردن نودهاي تكرار كننده در ميان شبكه، مي‌توان برد را گسترش داد، redundancy را افزود و اطمينان كلي شبكه را بهبود بخشيد.
۴-يك شبكه مش قابل گسترش است و مي تواند صدها يا هزاران نود را در برداشته باشد. استاندارد ZigBee تا ۶۵۵۳۶ نود شبكه را پشتيباني مي كند.
بر اساس ويژگي‌هاي فراهم شده و همچنين امنيت بالاي شبكه در تكنولوژي ZigBee (تكنولوژي ZigBee از الگوريتم AES ۱۲۸بيتي استفاده مي كند)، شبكه مش ZigBee در كاربرد قرائت خودكار كنتورها (AMR: Automatic Meter Reading) بسيار مناسب مي‌باشد.



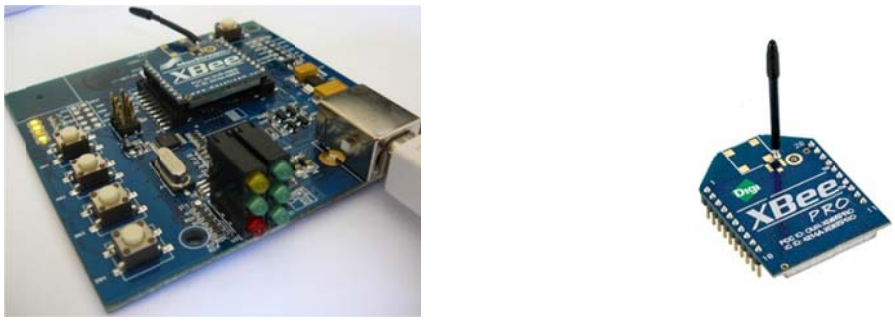
شكل۱-۱- توپولوژي‌هاي مختلف شبكه در تكنولوژي ZigBee

۱-۲- پياده سازي عملي

براي پياده‌سازي عملي اين استاندارد بي‌سيم، تراشه‌هاي متنوعي از سازندگان مختلف در دسترس مي‌باشد. تراشه‌اي كه در شركت مورد استفاده قرار گرفته است، XBee PRO Series2 ساخت شركت MaxStream مي‌باشد. مختصري به اين تراشه پرداخته مي‌شود و اطلاعات بيشتر را مي‌توان در ديتا شيت اين تراشه يافت.

۱-۲-۱- معرفي تراشه XBee PRO Series2

اين تراشه را در شكل۱-۲ مشاهده مي‌كنيد. اين تراشه را به سادگي مي‌توان در Starter Kit مربوطه قرار داد و با استفاده از پورت USB و يا RS232 (بسته به نوع Starter Kit) به كامپيوتر متصل نموده و آن را پيكربندي نمود.



شكل ۱-۲- تراشه XBee PRO Series2 و Starter Kit آن

این تراشه را می‌توان در دو حالت Transparent و API برنامه‌ریزی کرد. در حالت Transparent تراشه به صورت یک خط سریال عمل کرده و هر داده‌ای به پایه D_in آن ارسال شود را به صورت بی‌سیم ارسال می‌کند و اطلاعات RF دریافتی را به پایه D_Out انتقال می‌دهد. در حالت API که به دلیل برتری آن بر حالت Transparent در اغلب کاربردها مورد استفاده قرار می‌گیرد، اطلاعات در یک پکت با فرمت مشخص ارسال و دریافت می‌گردد. این پکت نوع عملکرد تراشه را تعیین می‌کند که می‌تواند تنظیم پارامتر، ارسال داده، دریافت داده و ... باشد. در هر یک از این دو حالت این تراشه می‌تواند به صورت Coordinator، Router و یا End Device برنامه ریزی گردد. به هنگام تشکیل یک شبکه ZigBee حتماً لازم است تنها یک Coordinator در شبکه وجود داشته باشد و سایر XBeeها بسته به نوع عملکردشان باید Router یا End Device باشند.

ساختار هر پکت بسته به نوع عملی که قرار است صورت پذیرد، متفاوت می‌باشد. ساختار کلی هر پکت API به صورت زیر می‌باشد:

مطابق این شکل، هر پکت API با 0x7E شروع می‌شود، پس از آن طول پکت، داده‌های مربوط به هر پکت و در نهایت نیز Checksum که به صورت (۸ بیت اول جمع بایت‌های پس از طول پکت -0xFF) می‌باشد، قرار می‌گیرد. در میان این پکت‌ها، ۴ پکت AT Command، AT Command Response، Transmit Request و Receive Packet پرکاربردتر و حائز اهمیت می‌باشد که به آن‌ها پرداخته می‌شود.

۱- AT Command

این دستور برای ست کردن و یا قرائت پارامترهای تراشه به کار می‌رود. به عنوان مثال پکت زیر برای ست کردن پارامتر NJ (Node Join Time) به کارگرفته می‌شود. لازم به ذکر است که هر نوع پکتی با یک شماره در قسمت Frame Type مشخص می‌شود و به عنوان مثال این شماره برای پکت AT Command، 0x08 می‌باشد.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x04	
	Frame-specific Data	Frame Type	3	0x08	
		Frame ID	4	0x52 (R)	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		AT Command	5	0x4E (N)	Command Name - Two ASCII characters that identify the AT Command.
			6	0x4A (J)	
		Parameter Value (optional)			If present, indicates the requested parameter value to set the given register. If no characters present, register is queried.
	Checksum		7	0x0D	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

۲- AT Command Response

در جواب هر AT Command، تراشه یک پاسخ ارسال می‌کند. در جواب برخی از AT Commandها نظیر ND (Node Discovery)، که برای کشف آدرس یک نود مورد استفاده قرار می‌گیرد، چند پکت ارسال می‌گردد.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x05	
	Frame-specific Data	Frame Type	3	0x88	
		Frame ID	4	0x01	Identifies the UART data frame being reported. Note: If Frame ID = 0 in AT Command Mode, no AT Command Response will be given.
		AT Command	5	'B' = 0x42	Command Name - Two ASCII characters that identify the AT Command.
			6	'D' = 0x44	
		Command Status	7	0x00	0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter
		Command Data			Register data in binary format. If the register was set, then this field is not returned, as in this example.
	Checksum		8	0xF0	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Transmit Request-۳

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x16	
	Frame-specific Data	Frame Type	3	0x10	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following addresses are also supported: 0x0000000000000000 - Reserved 64-bit address for the coordinator 0x000000000000FFFF - Broadcast address
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x0A	
			11	0x01	
			LSB 12	0x27	
		16-bit Destination Network Address	MSB 13	0xFF	Set to the 16-bit address of the destination device, if known. Set to 0xFFFE if the address is unknown, or if sending a broadcast.
			LSB 14	0xFE	
		Broadcast Radius	15	0x00	Sets maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius will be set to the maximum hops value.
		Options	16	0x00	Bitfield of supported transmission options. Supported values include the following: 0x20 - Enable APS encryption (if EE=1) 0x40 - Use the extended transmission timeout for this destination Enabling APS encryption decreases the maximum number of RF payload bytes by 4 (below the value reported by NP). Setting the extended timeout bit causes the stack to set the extended transmission timeout for the destination address. (See chapter 4.) All unused and unsupported bits must be set to 0.
		RF Data	17	0x54	Data that is sent to the destination device
			18	0x78	
			19	0x44	
			20	0x61	
			21	0x74	
			22	0x61	
			23	0x30	
			24	0x41	
	Checksum		25	0x13	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

با استفاده از این دستور می‌توان دیتای مورد نظر را در قالب یک پکت RF به مقصد خاص (یک ZigBee دیگر با آدرس مشخص) ارسال نمود. برای ارسال اطلاعات به کلیه XBeeها در یک شبکه آدرس ۶۴

بيتي مقصد را بايد 0x000000000000FFFF قرار داد. آدرس Coordinator را به دو صورت مي‌توان مشخص كرد. در روش اول آدرس ۶۴ بيتي را برابر 0x0 و آدرس ۱۶ بيتي را برابر 0xFFFE قرار مي‌دهيم. در روش دوم با دانستن آدرس ۶۴ بيتي واقعي Coordinator، در قسمت آدرس ۶۴ بيتي اين را قرار داده و آدرس ۱۶ بيتي Coordinator را 0x0000 قرار مي‌دهيم. در ساير ارسال‌ها در صورتيكه آدرس ۱۶ بيتي دانسته شده باشد، قرار دادن آن در پكت عملكرد را بهبود مي‌بخشد در غير اين صورت در قسمت آدرس ۱۶ بيتي 0xFFFE (دانسته نشده) قرار داده مي‌شود.

Receive Packet-۴

هنگامي كه XBee يك پكت RF دريافت كند، اين پكت با فرمت زير به UART مي‌فرستد.

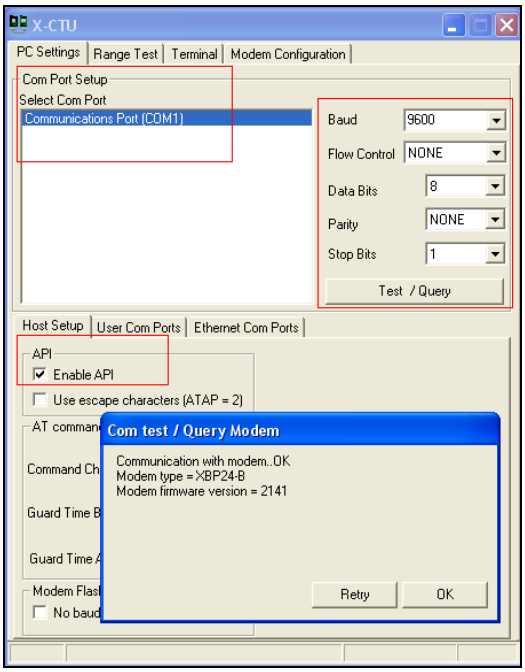
	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x11	
	Frame-specific Data	Frame Type	3	0x90	
		64-bit Source Address	MSB 4	0x13	64-bit address of sender. Set to 0xFFFFFFFFFFFFFFFF (unknown 64-bit address) if the sender's 64-bit address is unknown.
			5	0xA2	
			6	0x00	
			7	0x40	
			8	0x52	
			9	0x2B	
			LSB 10	0xAA	
		16-bit Source Network Address	MSB 11	0x7D	16-bit address of sender
			LSB 12	0x84	
		Receive Options	13	0x01	0x01 - Packet Acknowledged 0x02 - Packet was a broadcast packet 0x20 - Packet encrypted with APS encryption 0x40 - Packet was sent from an end device (if known)
		Received Data	14	0x52	Received RF data
			15	0x78	
			16	0x44	
			17	0x61	
			18	0x74	
			19	0x61	
	Checksum		20	0x0D	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

برنامه‌ريزي XBee ، ارسال و دريافت پكت از طريق آن، با استفاده از نرم‌افزار X-CTU كه براي تراشه‌هاي Xbee توسط شركت MaxStream نوشته شده است، انجام مي‌گيرد. در بخش بعدي به اين نرم افزار پرداخته مي‌شود.

۱-۲-۲- نرم‌افزار X-CTU

اين نرم افزار را مي‌توان از سايت www.digi.com دانلود كرده و به سادگي نصب نمود. در ادامه مختصري به نحوه كار با اين نرم-افزار پرداخته مي‌شود. براي اطلاعات بيشتر مي‌توان به راهنماي X-CTU مراجعه نمود. با استفاده از اين نرم افزار مي‌توان يك XBee خام را برنامه‌ريزي كرد، پارامترهاي يك XBee برنامه‌ريزي شده را تغيير داد، پكت‌هاي داده را از XBee ارسال كرده و نهايتاً پكت‌هاي دريافتي را مشاهده نمود.

براي ارتباط با يك XBee با استفاده از نرم‌افزار X-CTU بايد XBee را با استفاده از اين نرم‌افزار شناسايي كرد. بدين جهت پس از باز كردن X-CTU روي تب PC Setting كليك كرده و در قسمت Com Port Setup پورتي كه XBee با آن متصل است را انتخاب مي‌كنيم. پس از آن پارامترهاي ارتباط سريال از جمله Baud Rate و ... را در مطابق شكل۱-۳ در سمت راست پنجره X-CTU تعيين كرده و Enable API را فعال مي‌كنيم. سپس روي گزينه Test/Query كليك مي‌كنيم. با اين كليك، X-CTU مطابق شكل۱-۳ ، XBee را شناسايي مي‌كند. براي ادامه لازم است در پنجره شناسايي XBee روي Ok كليك نماييد.



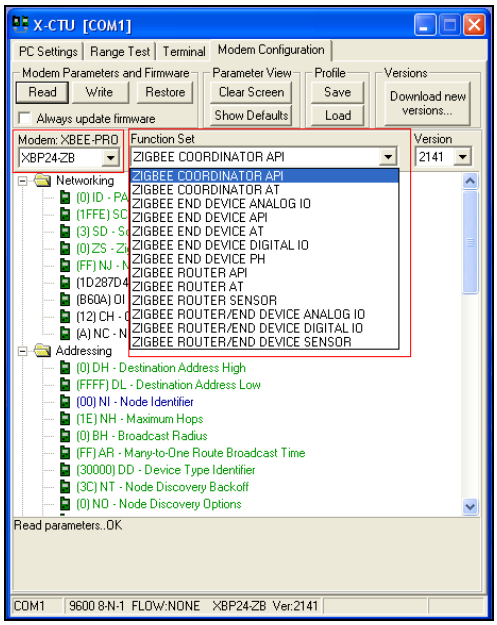
شكل ۱-۳- ارتباط با XBee با استفاده از نرم افزار X-CTU

برنامه‌ريزي XBee و يا قرائت پارامترهاي آن در تب Modem Configuration انجام مي‌شود. در صورتيكه XBee از قبل برنامه‌ريزي شده باشد با كليك بر روي Read تنظيمات آن مشخص مي‌گردد. در صورت نياز به برنامه‌ريزي XBee لازم است مراحل زير انجام گيرد (شكل۱-۴) :

۱-نوع XBee مورد استفاده در قسمت Modem تعيين مي‌گردد (براي XBee PRO Series 2 نوع آن XBP24-ZB مي‌باشد.)

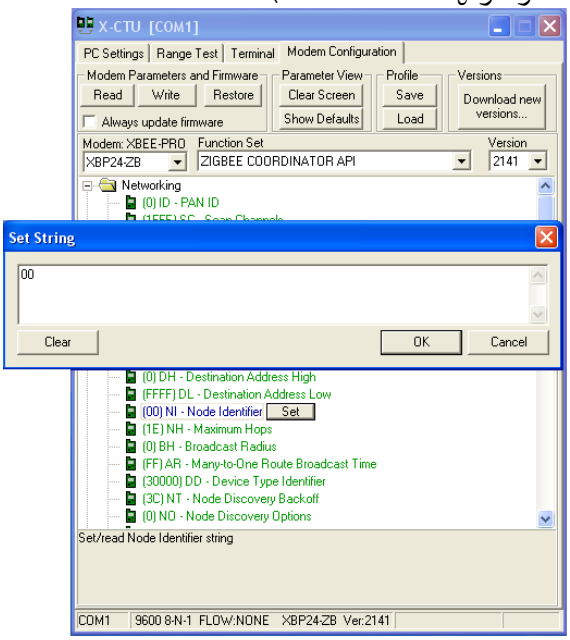
۲-نوع كاربري XBee ، نظير Coordinator API در قسمت Function Set تعيين مي‌گردد.

پس از تعيين اين دو پارامتر بر روي گزينه Write كليك كنيد تا اين تنظيمات بر روي XBee نوشته شود. اين تنظيمات در صورت قطع تغذيه نيز بر روي XBee باقي خواهد ماند.



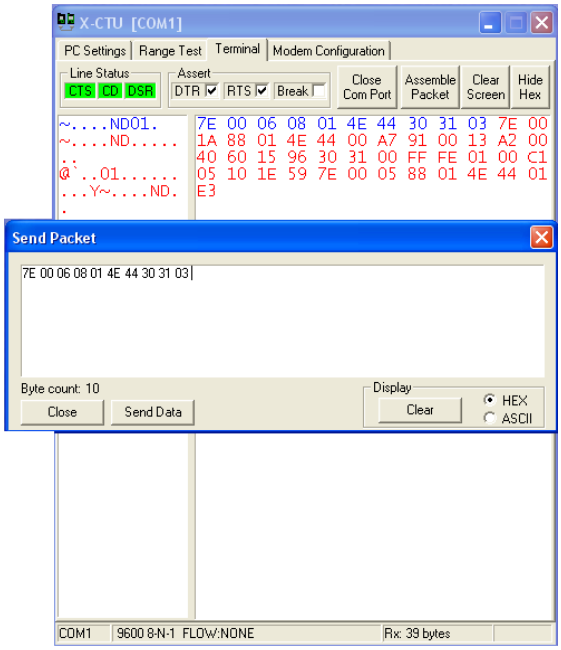
شكل ١-٤- برنامه‌ريزي XBee و مشاهده پارامترهاي آن

در تب Modem Configuration كليه پارامترهاي مربوط به XBee قابل تنظيم است. اين پارامترها بسيار متنوع هستند در اينجا به عنوان نمونه به تنظيم يكي از پركاربردترين آن‌ها يعني Node Identifier پرداخته مي‌شود. هر XBee داراي يك آدرس IEEE ۶۴ بيتي يكتا مي‌باشد. در ارتباطات بين نودها لازم است آدرس هر نود دانسته شده باشد. از آنجايي كه ثبت آدرس ۶۴ بيتي كليه نودها دشوار مي‌باشد مي‌توان به هر XBee يك شناسه نود (Node Identifier) تخصيص داده و در ارتباطات بين نودها از اين شناسه براي كشف آدرس ۶۴ بيتي نود و ارتباط با آن استفاده كرد (نحوه انجام اين كار در ادامه گزارش آمده است).



شكل ١-٥- تغيير پارامترهاي XBee

براي تنظيم اين پارامتر با كليك بر روي آن، مطابق شكل۱-۵ گزينه set كنار آن نشان داده مي‌شود كه با كليك روي اين گزينه مي‌توان شناسه مورد نظر را به آن اختصاص داد. به عنوان نمونه درشكل۱-۵ به Coordinator، شناسه 00 اختصاص داده شده است. پس از تغيير هر پارامتري جهت ثبت اين تغييرات روي XBee لازم است بر روي گزينه write كليك نماييد.



شكل ۱-٦- ارسال پكت و مشاهده پكت دريافتي

براي ارسال يك پكت داده و يا مشاهده پكت داده دريافتي مطابق شكل۱-۶ از تب Terminal استفاده مي‌شود. براي مشاهده معادل هگزادسيمال پكت‌هاي ارسالي و دريافتي لازم است روي گزينه Show Hex در گوشه سمت راست كليك نماييد. در اين صورت هرگاه پكتي توسط XBee متصل به كامپيوتر دريافت گردد در قسمت سمت راست معادل هگزادسيمال آن و در قسمت سمت چپ معادل ASCI آن نمايش داده مي‌شود. براي ارسال يك پكت لازم است بر روي گزينه Assemble Packet كليك كرده و پس از نوشتن كل پكت با كليك بر روي Send Data مطابق شكل۱-۶ پكت را ارسال نماييد. به عنوان نمونه در شكل۱-۶ پكت Node Discovery كه با استفاده از آن آدرس ۶۴ بيتي نود مقصد با دانستن Node Identifier آن كشف مي‌شود، به يك نود با Node Identifier=01 ارسال شده (پكت آبي رنگ) و در جواب پاسخ آن نود و در نتيجه آدرس ۶۴ بيتي آن يعني 00 13 A2 00 40 60 15 96 دريافت شده است (پكت قرمز رنگ).

فصل دوم:

قرائت از راه دور کنتورهاي ديجيتال با استفاده از شبکه-هاي مش ZigBee

۲-۱- معرفي روش قرائت خودکار کنتور

در حال حاضر کنتورهای برق، آب و یا گاز با مراجعه ماهانه مأمور به محل نصب کنتورها قرائت می شود. با توجه به این که

اغلب □نتورها در داخل منازل شخصی نصب هستند، اگر مصرف □ننده در خانه نباشد مأمور نمی تواند مصرف مشترک را قرائت □ند. عدم رضایت برخی مشتر□ین در مورد ورود مأموران قرائت □نتور به داخل منازل شخصی، استخدام نیروی انسانی، عدم امکان اندازه گیری مصرف لحظه ای مشتر□ین و در نتیجه عدم امکان مدیریت توزیع توان از دیگر مشکلات روش موجود است. به منظور رفع این مشکلات تلاشهایی برای قرائت هوشمند □نتور (AMR) در جهان در حال انجام است.
روش قرائت هوشمند کنتورها عبارت از فن‌آوري خود□ار جمع‌آوري اطلاعات از کنتورهايي چون کنتور ديجيتالي آب، برق و يا گاز، انتقال آن داده ها به مرکز براي تجزيه و تحليل آن داده ها و صدور صورتحساب براي مشترکان می باشد.
تکنولوژي اندازه گيري هوشمند قادر است مصرف انرژي و ديگر پارامترهاي اندازه گيري را ذخيره کرده و در فواصل زماني دلخواه گزارش دهد. اين قرائت زمان واقعي به مصرف کنندگان اين امکان را مي دهد که از ميزان مصرف انرژي و هزينه پرداختي به طور لحظه اي مطلع شوند و مصرف انرژي خود را متناسب با آن بخصوص در ساعات اوج مصرف کاهش دهند. از طرفي در مراکز انرژي علاوه بر کاهش هزينه هاي ناشي از حذف نياز به قرائت دستي کنتورها، اطلاعات زمان واقعي از چگونگي مصرف انرژي و در نتيجه امکان مديريت توزيع توان فراهم مي‌شود.

۲-۲- کنتور ديجيتال Elster A220 و نرم افزار AlphaSET

در پيادهسازي پايلوت AMR از کنتور Elster A220 استفاده شده است که در ادامه مختصري به ويژگيهاي آن پرداخته مي‌شود. اين کنتور امنيت بسيار بالا، دقت زياد و ساختار تعرفه‌بندي جامعي دارد که مناسب براي اندازه‌گيري مصارف خانگي مي‌باشد. اين کنتور امکان گزارش‌دهي لودپروفايل را تا ۴۲۰ روز در اختيار قرار مي‌دهد.
ارتباط با اين کنتور از طريق يك پورت نوري مطابق با استاندارد IEC61107 (يا همان IEC62056-21) ميسر مي‌باشد. اين پورت نوري از يك طرف بر روي پورت نوري کنتور قرار گرفته و از طرف ديگر به پورت سريال کامپيوتر متصل مي‌گردد و سپس توسط يك نرم-افزار تحت ويندوز به نام alphaSET مي‌توان کنتور را برنامه‌ريزي و پيکربندي کرده و مقادير اندازه‌گيري شده را قرائت نمود.
اين کنتور دربرگيرنده دامنه وسيعي از داده‌هاي امنيتي (اطلاعات حفاظت شده)، همراه با شاخص‌هاي نشان دهنده و تاريخ وقوع آن مي‌باشد. اين موارد شامل:

- شمارش دفعات معکوس شده
- مجموع انرژي برگشتي
- نمايش دهنده کارکرد معکوس
- ساعات کارکرد دستگاه
- تعداد دفعات قطع برق
- ساعات خارج بودن دستگاه از مدار (استفاده غير مجاز از برق)
- دفعات برنامه‌ريزي دستگاه

- امكان نمايش صورت حسابها (محاسبه نوبتهاي صدور قبض)
- دفعات ريست (Reset) دستگاه (صفركردن)
- زمان وتعداد برداشتن پوشش ترمينال

اين كنتور در شكل۲-۱ نشان داده شده است و براي اطلاعات بيشتر در مورد آن مي‌توانيد به ديتاشيت كنتور مراجعه نماييد.

شكل ۲-۱- كنتور Elster A220 و كابل پرت نوري براي ارتباط سريال با آن

در ادامه مختصري به نرم‌افزار alphaSET پرداخته مي‌شود. اين نرم-افزار بسيار گسترده بوده و در ادامه تنها نحوه قرائت رجيسترهاي آن شرح داده مي‌شود. براي مطالعه بيشتر مي‌توان به كتابچه اين نرم افزار مراجعه نمود. براي قرائت رجيسترهاي كنتور توسط alphaSET لازم است مراحل زير انجام گردد:

۱-پس از باز كردن نرم افزار پنجره Select User-Profile ظاهر مي‌شود كه در آن مطابق شكل ۲-۲ الف لازم است Meter Reader انتخاب گردد.

۲-پس از باز شدن صفحه اصلي نرم افزار در تب Meter مطابق شكل ۲-۲ ب گزينه Read Register انتخاب گردد.

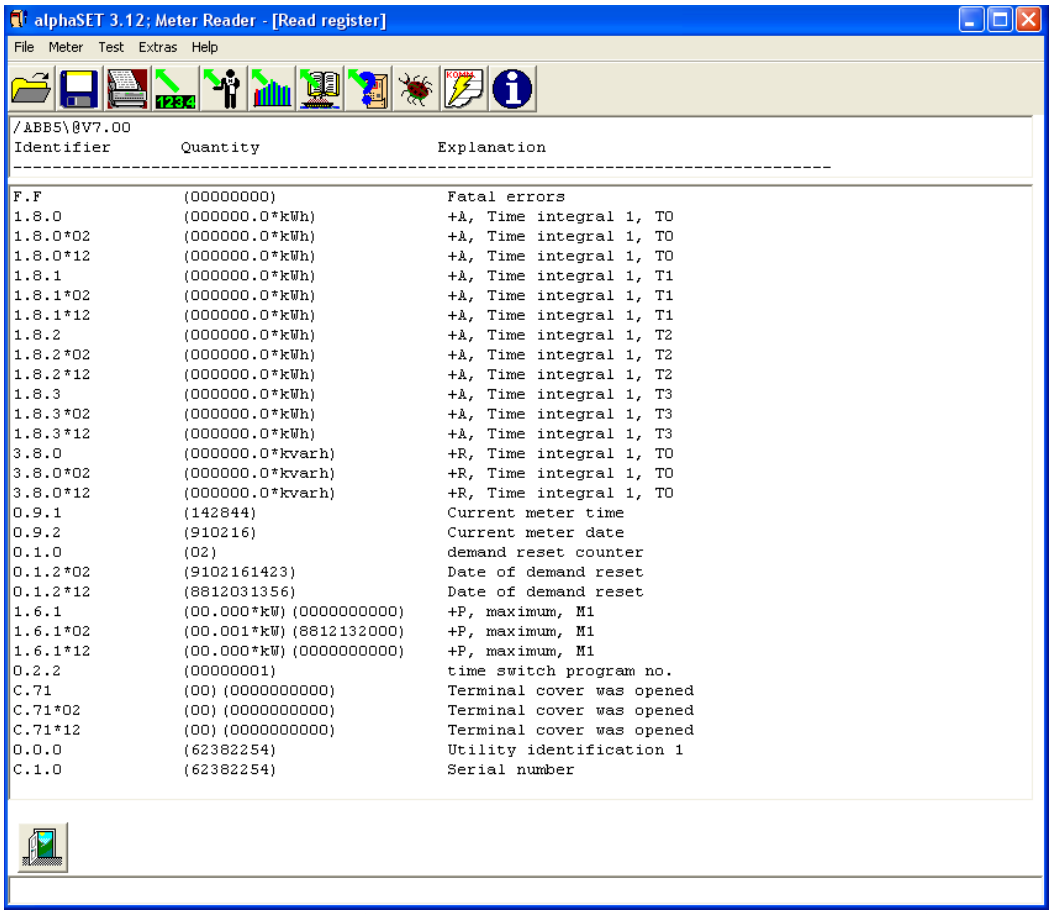
۳-مقادير مربوطه قائت شده و مانند شكل۲-۳ در جدولي شامل Identifier، Quantity و Explanation نشان داده مي‌شود.

جهت پيادەسازي سيستم قرائت خودكار كنتورها لازم بود از جزئيات پكت‌هاي مبادله شده بين كنتور و پورت سريال كامپيوتر مطلع گرديم. بدين جهت به هنگام اجراي نرم‌افزار alphaSET نرم افزار Serial Port Monitor را اجرا نموديم تا پكت‌هاي مبادله شده در پورت سريال كامپيوتر را به ما نشان دهد. شكل۲-۴ جزئيات اين پكت‌ها را نمايش مي‌دهد.

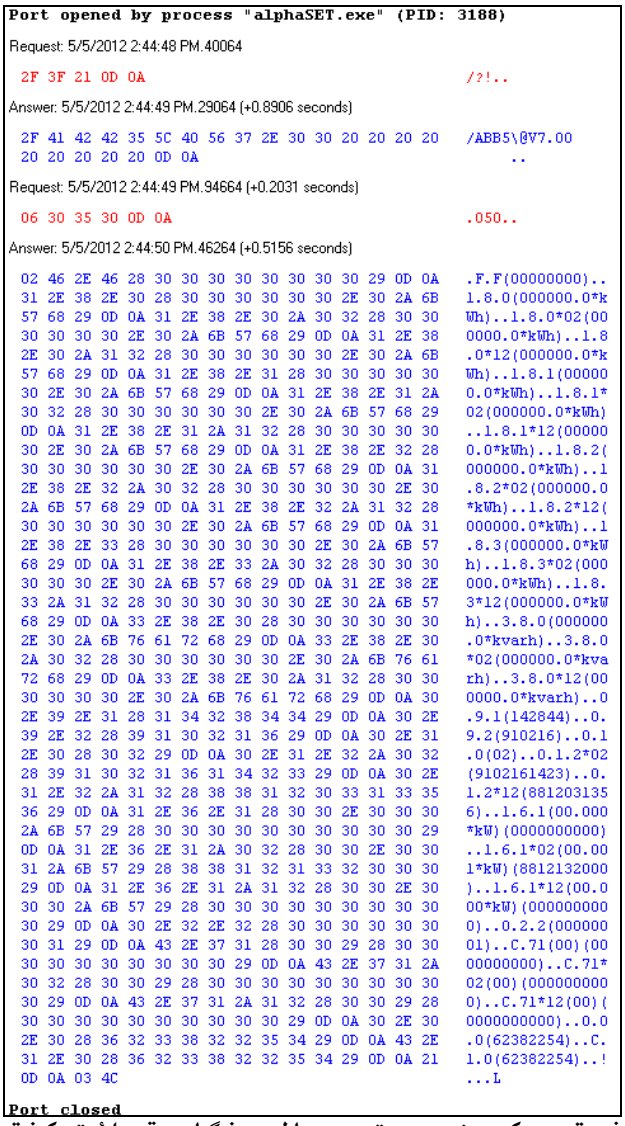
(الف)

(ب)

شکل ۲-۲- ورود به نرم افزار **alphaSET** جهت قرائت رجیسترهای کنتور



شکل ۲-۳-نتیجه قرائت کنتور با نرم افزار **alphaSET**



شكل ٢-٤-نتيجه مونيتور كردن پورت سريال هنگام قرائت كنتور با alphaSET

بر اساس نتايج حاصل از مونيتور كردن پورت سريال پكت‌هاي ارسالي به كنتور و دريافتي از آن به صورت زير مي‌باشد:
ويژگي‌هاي پورت سريال در ابتداي قرائت به صورت زير است:

- Baud rate: 300 bits/sec
- Data bits: 7
- Parity: even

ابتدا نرم افزار با ارسال پكت زير روي كنتور sign on مي‌كند:

بايت ارسالي	2F	3F	21	0D	0A
كاراكتر معادل بايت	/	?	!	\r	\n
توضيح	Start character	Request command	End character	Completion characters	

در پاسخ به اين ارسال، كنتور به صورت زير پاسخ مي‌دهد:

بايت	2F	41	42	42	35	5C	40	56	37	2E

بايت ارسالي										
كاراكتر معادل بايت	/	A	B	B	5	\	@	V	7	.
توضيح	Start character	Manufacturer identification			Baud rate	Identification characters				

بايت ارسالي	30	30	20	20	20	20	20	20	20	20	20
توضيح	Identification characters										

بايت ارسالي	0D	0A
توضيح	Completion characters	

سپس نرم افزار پكت زير را جهت درخواست قرائت به كنتور ارسال مي‌كند:

بايت ارسالي	06	30	35	30	0D	0A
كاراكتر معادل بايت	.	0	5	0	\r	\n
توضيح	Ack character	Protocol control	Baud rate	Mode control	Completion character	

سپس نرم افزار ۳۰۰ms صبر مي‌كند، سرعت را به ۹۶۰۰bits/sec تغيير داده و پاسخ كنتور را كه همان مقادير رجيسترهاست، دريافت مي-كند.

۲-۳-روش AMR پيشنهادي

در روش پيشنهادی برای قرائت هوشمند كنتور از شبكه مش بی سيم zigbee استفاده مي شود كه نسبت به ساختارهاي رايج از جهات مختلف برتري دارد. ويژگيهايي نظير مصرف توان بسيار پايين، امنيت بالای شبكه، قابليت اطمينان بالا، انعطاف پذيری، قابليت گسترش، استفاده از باند فركانسي آزاد، ارتباط با برد نسبتاً كوتاه، سازماندهی، شكل دهی و التيام خودكار شبكه، هزينه كم، سهولت نصب و ... اين تكنولوژی را براي مديريت و كارايي انرژي ايده آل مي سازد.

سيستم پيشنهادی از ارتباط بين نودهای AMR در شبكه های مش بهره می گيرد تا داده ها را به مراكز جمع آوری كننده برساند. از آنجايی كه كنتورها در مناطق مسكونی از هم دور نيستند، هر نود AMR قادر است داده ها را از طريق ارتباط بی سيم با برد نسبتاً كوتاه با واسطه نودهای همسايه اش به طرف مركز جمع آوری داده بفرستد.

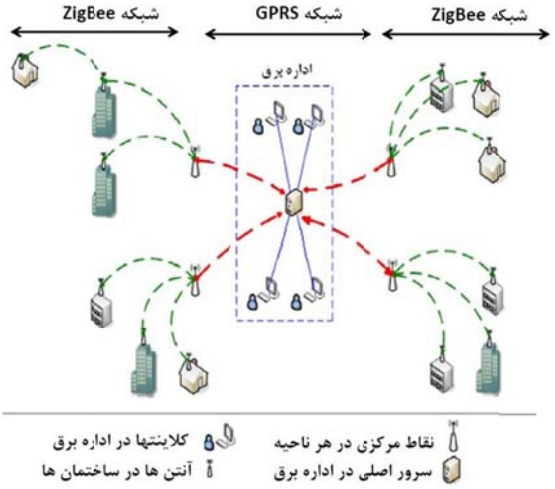
برخی از مهمترين مزايای روش پيشنهادی عبارتند از:

- عدم استفاده از نيروي انساني و دخالت دستي در امر قرائت كنتور

- امكان قرائت لحظه اي □نتورهاي ديجيتالي برق به صورت، روزانه ، هفتگي و ماهانه و يا هر در هر فاصله زماني □ه □ارفرما بخواهد. سامانه ياد شده قابل تنظيم و برنامه ريزي مي باشد. (اين سامانه مي تواند در□نتورهاي ديجيتالي آب و گاز نيز □اربرد داشته باشد)
- اطمينان □امل از امنيت و صحت اطلاعات دريافتي ناشي از قرائت و جلوگيري از خطاهاي انساني
- قرائت چندين كنتور، مثلاً در يك مجتمع مسكوني تنها با يك ماژول Zigbee مي توان اطلاعات تمامي كنتورهاي مجتمع را دريافت كرد
- انتقال اطلاعات قرائت □نتور ها به صورت اتوماتيک به سامانه مر□زي با دقت و با قبوليت داده ها
- امكان قطع و وصل برق، خاموش و روشن □ردن □نتورها با توجه به بدهي مشترک و يا به هر دليل ديگر. اين عمل بدون حضور مأمور در محل نصب □نتور صورت مي گيرد.
- امكان تنظيم دستگاه Zigbee بدون حضور مأمور در محل نصب □نتور (تنظيم از راه دور)
- امكان اعمال تغييرات در تنظيمات □نتور ها در يک زمان □وتاه
- امكان تبادل اطلاعات دو طرفه بين مر□ز و □نتورهای برق
- امكان تغيير مسير ارسال اطلاعات به هرسروري يا مر□ز جمع آوري داده ها (Data Server) □ه به سامانه معرفي مي‌گردد
- جلوگيري از هر گونه دستكاري در پلمپ □نتورهاي ديجيتال نصب شده در محل
- با توجه به قرائت به صورت روزانه، هفتگي و . . . علاوه بر □نترل مصرف، گرفتن آمار مصرف برق و يا صدور زود هنگام صورت حساب برای مشتر□ان امكان پذير است
- صدور صورت حساب به مشتركين از طريق SMS
- هشدار به مشتركين در ساعات اوج مصرف
- استفاده از شبكه موبايل موجود در مناطق تحت پوشش و نيز امكان پياده سازي در مناطقي كه تحت پوشش موبايل نيستند.
- امكان ارسال اطلاعات به صورت پيام □وتاه (SMS) ويا استفاده از بستر GPRS براي ارسال انواع داده ها (تصوير و صدا)
- بومي بودن طراحي و ساخت سخت افزار و نرم افزار هاي مربوطه
- عدم مراجعه مأمور به درب خانه مشتر□ان و جلوگيري از ورود افراد به حريم خصوصي مردم اين امر موجب بالا رفتن امنيت در جامعه مي‌گردد.

در شبكه AMR پيشنهادي كه شماتيک آن در شكل۲-۵ مشاهده مي‌شود، از تركيب شبكه‌هاي مش ZigBee و شبكه GPRS استفاده شده است. برای نمونه اين سيستم برای □نتورهای برق تشريح می گردد، درحاليكه اين تكنولوژی برای هر □دام از □نتورهای ديجيتال آب، برق و يا گاز قابل اعمال است.

هدف اصلی در این طراحی، ارسال قرائت های پریودیک از □نتورهای برق به یک سرور در اداره برق است. در این طراحی □ل منطقه به نواحی جغرافیایی □وچکتر تقسیم می شود. هر □دام از این نواحی دارای یک نود مر□زی برای جمع آوری داده های قرائت شده از □نتورهای آن ناحیه می باشد. □نتورهای موجود در هر ناحیه با یک شبکه مش ZigBee به یکدیگر متصلند و داده های قرائت شده خود را از طریق این شبکه به نود مر□زی در ناحیه خود ارسال می نمایند. در نقاط مرکزي، با استفاده از یک ریزپردازنده و یک مودم GPRS، داده های جمع آوری شده در بستر GPRS به سرور اداره برق منتقل مي‌گردند.



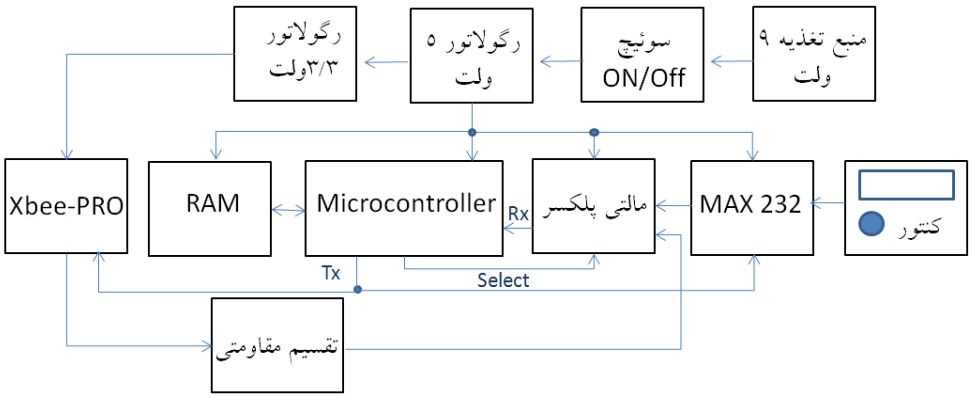
شکل ۲-۵- شمای کلی سیستم AMR با استفاده از شبکه های مش ZigBee

۲-۳-۱-طراحي و پیاده سازي سخت افزار

شبکه AMR پیشنهادي از سه قسمت اصلی تشکیل شده است: ۱- بخش فرستنده (متصل به □نتورها) ۲- بخش نقاط مر□زی ۳- بخش گیرنده (سرور اداره برق).

الف- بخش فرستنده

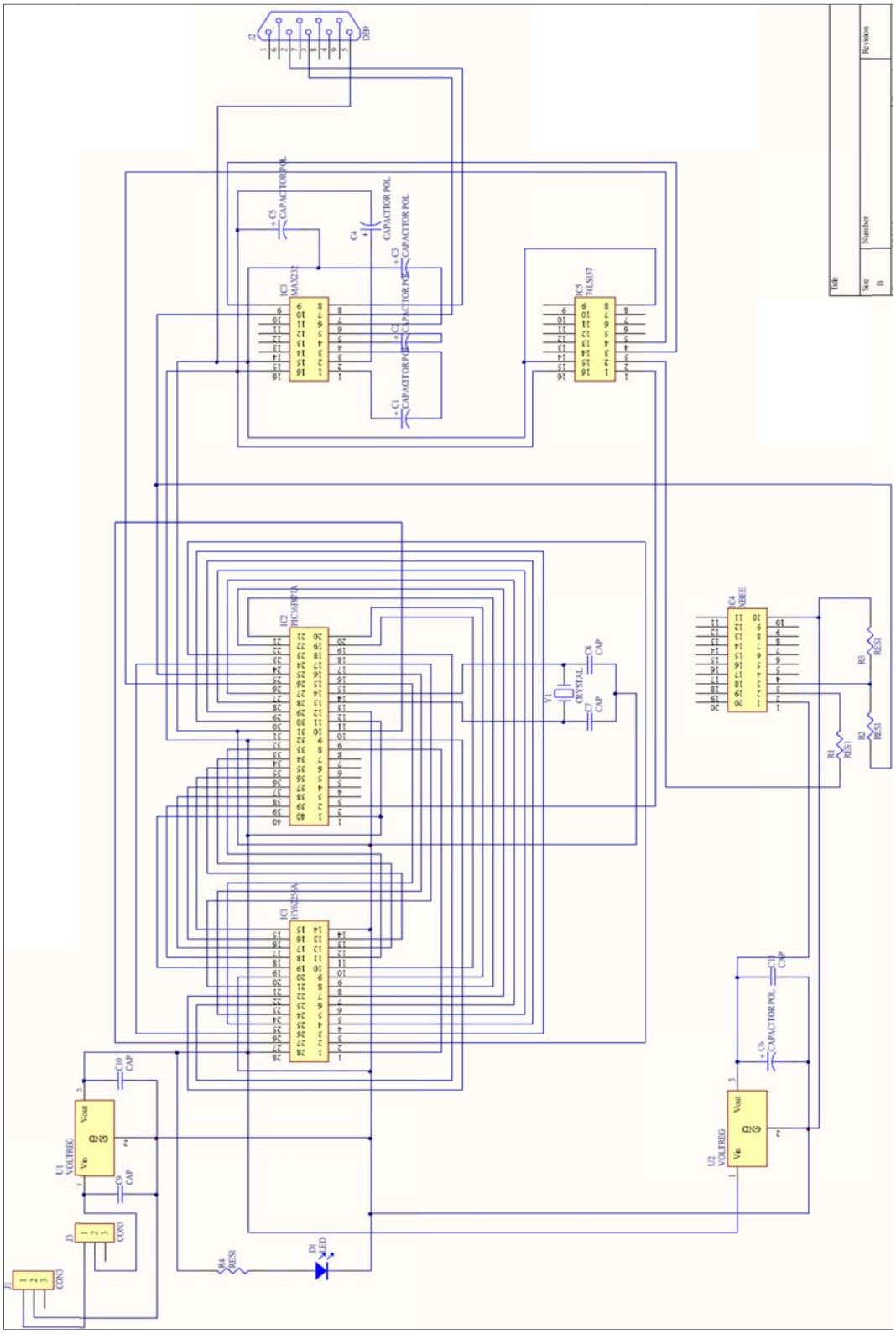
بخش فرستنده شامل یک □نتور دیجیتال جهت قرائت، یک میکرو□نترلر برای □نترل قرائت ، یک حافظه RAM غیر فرار برای ذخیره □ردن اطلاعات □نتور و یک چیپ ZigBee جهت ارسال بی سیم داده ها به نقاط مر□زی می باشد. در این بخش، پکت ارسالي از نود مرکزي توسط چیپ ZigBee دریافت شده و توسط میکروکنترلر پردازش مي‌شود. بسته به پیام ارسالي، میکروکنترلر دستورات لازم را براي کنتور ارسال کرده و اطلاعات دریافتي از کنتور را در حافظه RAM ذخیره مي‌کند. پس از اتمام ذخیره‌سازي اطلاعات، این داده‌ها براي ارسال به چیپ ZigBee منتقل شده تا از طریق آن به صورت بي‌سیم به نود مرکزي ارسال گردند. بلوك دیاگرام مدار بخش فرستنده و جریان اطلاعات در آن در شکل۲-٦ آمده است.



شکل ۲-۶- بلوک دیاگرام بخش فرستنده و جریان اطلاعات در آن

چیپ ZigBee مورد استفاده در این پیاده سازي چیپ XBee PRO Series2 ساخت شرکت MaxStream بوده که داراي توان اندك و قابلیت اطمینان بالا مي‌باشد. برد آن براي ارسال و دریافت اطلاعات در حدود ۴۰ متر در ساختمان‌ها و ۱۲۰ متر در فضاي باز مي‌باشد. این چیپ از تغذیه ۳/۳ ولتي استفاده مي‌کند و جریان ارسال و دریافت آن در این ولتاژ تغذیه برابر ۴۰ میلي آمپر است. نرخ ارسال اطلاعات آن ۲۵۰ کیلوبیت بر ثانیه و توان مصرفي آن ۲ میلي وات مي‌باشد. براي انتقال نود به نود اطلاعات بي‌سیم، چیپ ZigBee در بخش فرستنده لازم است به صورت Router برنامه‌ریزي گردد. میکروکنترلر مورد استفاده از نوع PIC16F887 است. حافظه RAM استفاده شده، یك حافظه ۳۲ کیلوبایتي از نوع HY62256A مي‌باشد که از ویژگي‌هایي نظیر سرعت بالا و توان اندك برخوردار است. یك ملتي پلکسر74LS157 نیز روي برد تعبیه شده است که تعیین مي‌کند که مبدا ارسال اطلاعات و یا مقصد دریافت اطلاعات کنتور دیجیتال و یا چیپ ZigBee باشد.

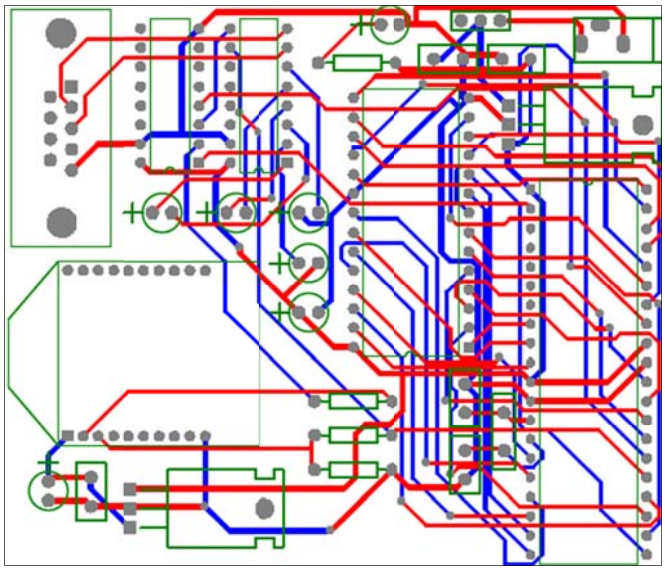
این مدار از یك آداپتور ۹ ولتي براي تغذیه استفاده مي‌کند. این ولتاژ تغذیه ابتدا وارد رگولاتور ۵ ولتي مي‌گردد تا ولتاژ مورد نیاز براي کلیه چیپ‌هاي قرار گرفته روي برد را تامین کند. از آنجایي که XBee-PRO با تغذیه ۳/۳ ولتي کار مي‌کند، ولتاژ خروجي رگولاتور ۵ ولتي وارد رگولاتور ۳/۳ ولتي مي‌شود تا سطح ولتاژ مورد نیاز XBee را فراهم آورد. این برد باید قادر باشد به هنگام دریافت بي‌سیم دستور قرائت کنتور، رجیسترهاي کنتور را قرائت کرده و به صورت بي‌سیم به مرکز جمع آوري اطلاعات کنتور ها ارسال کند. پردازش در این برد به وسیله میکروکنترلر انجام مي‌گیرد. بنابراین میکروکنترلر باید قادر به برقراري ارتباط با XBee و نیز با کنتور باشد. بدین ترتیب میکروکنترلر با استفاده از یك مالتي پلکسر تعیین مي‌کند اطلاعات از کدام منبع (کنتور یا XBee) وارد میکروکنترلر گردد. از آنجایي که سطح ولتاژ میکروکنترلر با XBee (به ترتیب ۵ و ۳/۳ ولت) متفاوت است، از یك مقسم ولتاژ مقاومتي در مدار استفاده شده است. MAX232 نیز براي تطبیق سطوح ولتاژ در ارتباط سریال با کنتور استفاده شده است. شماتیك مدار، BOM، طرح PCB مدار و نهایتاً مدار ساخته شده به ترتیب در شکل‌هاي ۲-۷، ۲-۸ ، ۲-۹ و ۲-۱۰ نشان داده شده است.



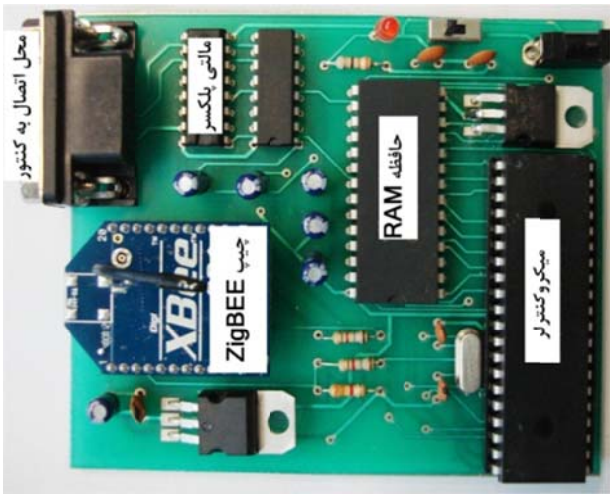
شکل ۲-۷- طرح PCB مدار بخش فرستنده

Comment	Description	Designator	Footprint	LibRef	Quantity
CAPACITOR POL	Capacitor	C1, C2, C3, C4, C5, C6	RB.1/.2	CAPACITOR POL	6
CAP	Capacitor	C7, C8, C9, C10, C11	RAD0.15	CAP	5
LED		D1	LED	LED	1
HY62256A		IC1	DIP28	HY62256A	1
PIC16F877A		IC2	DIP40	PIC16F877A	1
MAX232		IC3	DIP16	MAX232	1
XBEE		IC4	XBEE	XBEE	1
74LS157		IC5	DIP16	74LS157	1
CON3	Connector	J1	ADAPTOR	CON3	1
DB9		J2	DB9/M	DB9	1
CON3	Connector	J3	TO-126	CON3	1
RES1		R1, R2, R3, R4	AXIAL0.4	RES1	4
VOLTREG		U1, U2	TO-220	VOLTREG	2
CRYSTAL	Crystal	Y1	XTAL1	CRYSTAL	1

شکل ۲-۸- BOM المان‌هاي نشان داده شده در شکل ۲-۷



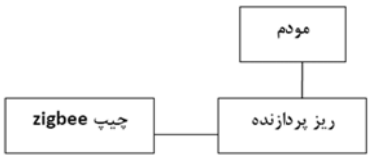
شکل ۲-۹- طرح PCB مدار بخش فرستنده



شکل ۲-۱۰- مدار ساخته شده براي بخش فرستنده

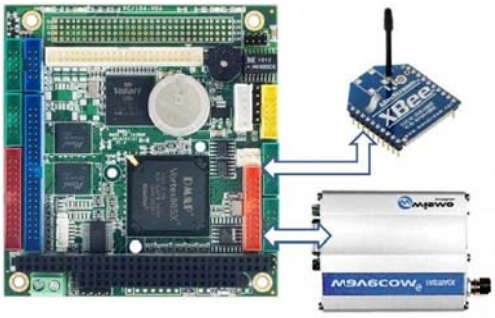
ب- بخش نقاط مرکزی(Collector)

بلوك نقاط مركزي در شكل۲-۱۱ نشان داده شده است. اين بخش شامل يک چيپ ZigBee، يک ريزپردازنده و يک مودم است که داده های جمع آوری شده از نتورهای آن منطقه را در بستر GPRS به سرور اداره برق منتقل می کند. ريزپردازنده بسته به فرمان مورد نظر (به عنوان مثال فرمان قرائت کنتور و يا تغيير پارامتري روي کنتور) و نيز آدرس کنتور مورد نظر، پکتي را به صورت بي‌سيم به بخش فرستنده ارسال نموده و پس از آن اطلاعاتي را که بخش فرستنده به آن ارسال مي‌کند، دريافت مي‌کند. پس از دريافت اطلاعات مذکور، با استفاده از مودم GPRS آن‌ها را به سرور مورد نظر ارسال مي‌کند.



شکل ۲-۱۱- بلوک دیاگرام طراحی سخت افزار بخش نقاط مرکزي

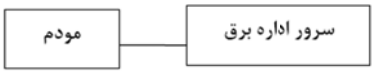
چيپ ZigBee در نقطه مرکزي لازم است به صورت Coordinator برنامه‌ريزي گردد که وظيفه هماهنگي شبکه را بر عهده دارد. مودم مورد استفاده در اين طراحي Wavecom Fastrack M1306B مي‌باشد. به عنوان ريزپردازنده مي‌توان از يک کامپيوتر استفاده نمود. پس از اطمينان از عملکرد سيستم در ساير کاربردها در صورت محدوديت فضا مي‌توان از يک Embedded Computer به عنوان ريزپردازنده استفاده نمود (برد VSX–6154-V2 بوده که يک Embedded Computer از نوع PC/104 مي‌باشد در شرکت خريداري شده و قابل استفاده مي‌باشد، تحقق اين طراحي در شکل۲-۱۲ قابل مشاهده مي‌باشد). در اولين مرحله پياده‌سازي براي سادگي کار مي‌توان از يک کامپيوتر متصل به اينترنت ADSL استفاده کرد.



شکل ۲-۱۲- سخت افزار به کارگرفته شده در بخش نقاط مرکزي

پ- بخش گيرنده

بلوك ديا‌گرام بخش گيرنده در شكل۲-۱۳ نشان داده شده است. بخش گيرنده شامل يک مودم است که به سرور اداره برق متصل می باشد. سرور با استفاده از اين مودم، به مودم متصل به ريزپردازنده در بخش نقاط مرکزی متصل شده و پس از برقراری ارتباط، انتقال اطلاعات از نقاط مرکزی به سرور آغاز می شود.



شکل ۲-۱۳- بلوک دیاگرام طراحی سخت افزار بخش گیرنده

۲-۳-۲-طراحي و پياده سازي نرم افزار

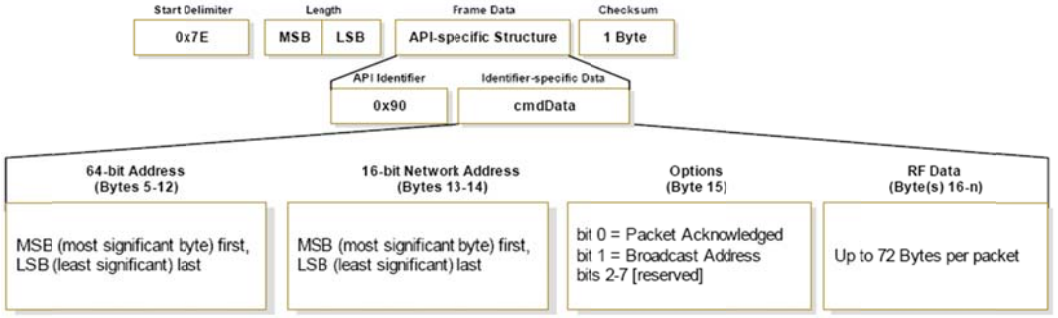
بخش نرم افزار شامل برنامه میکروکنترلر مدار گیرنده و نیز نرم افزار سیستم AMR میباشد که در ادامه به آن پرداخته می-شود.

الف-برنامه میکروکنترلر در مدار بخش گیرنده

شرح برنامه

این برنامه شامل مراحل زیراست:

مرحله اول: در این برنامه ابتدا میکروکنترلر منتظر دریافت پکت درخواست قرائت از طرف Collector میماند. این پکت ساختاري مطابق شکل۲-۱۴ دارد (ساختار Receive Packet).



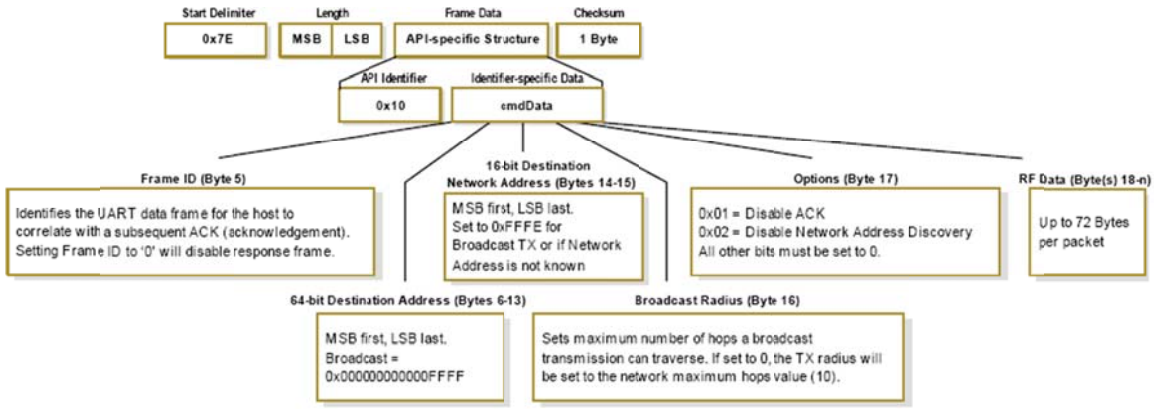
شکل ۲-۱۴-ساختار Receive Packet

در این طراحي بخش RF Data یك بایت 00 میباشد. بنابراین پکت مورد نظر در این قسمت به صورت زیر است:

7E	00	0D	90	00	13	A2	00	40	3	CC	D9	FF	FE	01	00	9D
				Collector IEEE Address											Data	

مرحله دوم: گر این پکت دریافت شد، میکروکنترلر دستورات لازم براي قرائت کنتور را صادر کرده و داده‌هاي قرائت شده از کنتور را در RAM خارجي ذخیره میکند

مرحله سوم: در این مرحله میکروکنترلر داده‌هاي ذخیره شده در RAM را میخواند و براي ارسال به Collector به XBee روي برد ارسال . از آنجایي که حجم رجیسترها؛ کنتور بیش از گنجایش یك پکت XBee میباشد، لذا میکروکنترلر داده‌هاي کنتور را طي چندین پکت به Collector ارسال میکند. این پکت‌ها بر اساس ساختار پکت Transmit Request مطابق شکل۲-۱۵ میباشند.

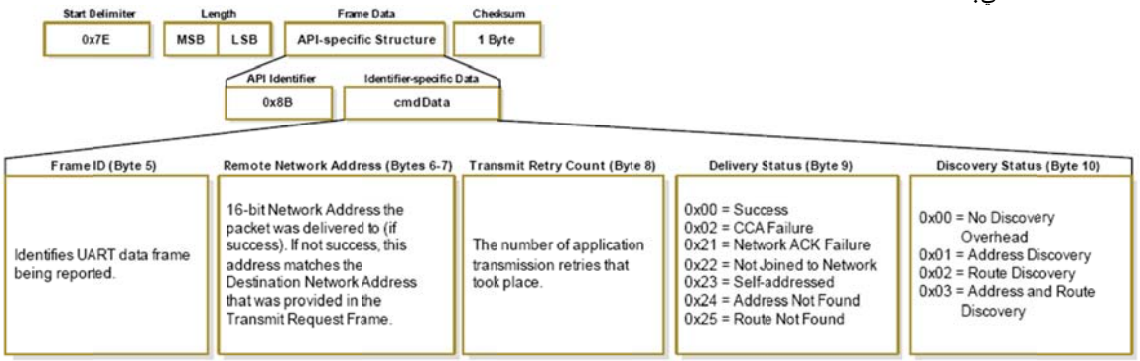


شکل ۲-۱۵-ساختار Transmit Request

بر این اساس این پکت مطابق زیر است:

7E	00	56	10	01	00	13	A2	00	40	3A	CC	D9	FF	FE	00	00	00	…	00	1D
					Collector IEEE Address												با فرض ۷۲ بایت داده‌ی صفر			

مرحله چهارم: با ارسال هر یک از این پکت‌های داده منتظر دریافت وضعیت ارسال آن (آیا پکت با موفقیت به Collector رسیده یا شکست خورده است) می‌نشیند. این پکت بر اساس ساختار Transmit Status مطابق شکل ۲-۱۶ می‌باشد.



شکل ۲-۱۶-ساختار Transmit Status

بر اساس این ساختار، پکت مورد نظر به صورت زیر است:

7E	00	07	8B	01	FF	FE	00	00	00	76
								Success		

اگر پکت وضعیت ارسال نشان دهنده شکست ارسال باشد یا انتظار برای دریافت پکت وضعیت ارسال timeout شود، دیگر میکروکنترلر پکت داده بعدی را ارسال نمی‌کند و منتظر می‌ماند تا دوباره پکت جدید درخواست قرائت کنتور از جانب Collector دریافت گردد.

مرحله پنجم: پس از اینکه تمام داده‌های قرائت شده از کنتور با موفقیت ارسال شد، منتظر می‌ماند تا پکت درخواست قرائت کنتور مجدداً از جانب Collector دریافت گردد و برنامه از ابتدا پیگیری می‌شود.

نحوه اختصاص پورت‌های میکروکنترلر

در جداول زیر نحوه ارتباط میکروکنترلر و تراشه RAM (HY62256A) ارائه شده است.

پورت میکروکنترلر	B0	B1	B2	B3	B4	B5	B6	B7
پایه HY62256A (پایه دیتا)	I/O1	I/O2	I/O3	I/O4	I/O5	I/O6	I/O7	I/O8

پورت میکروکنترلر	D0	D1	D2	D3	D4	D5	D6	D7	C0	C1	C2	C3	C4	C5	E0
پایه HY62256A (پایه آدرس)	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14

پورت میکروکنترلر	E2	E1
پایه HY62256A (پایه دیتا)	$\overline{WE}$ (Write Enable)	$\overline{OE}$ (Output Enable)

در جدول زیر نیز نحوه Read و Write در RAM آورده شده است.

Write نحوه	**Read** نحوه
$\overline{WE} = 1$	$\overline{WE} = 1$
$\overline{OE} = 1$	فراهم آوردن آدرس روي A0 تا A14
فراهم آوردن آدرس روي A0 تا A14	$\overline{OE} = 0$
فراهم آوردن ديتا روي I/O1 تا I/O8	فراهم آوردن ديتا روي I/O1 تا I/O8
$\overline{WE} = 0$	
$\overline{WE} = 1$	

پايهيSelect مالتي پلكسر نيز به پايه شماره0 از پورتA ميكروكنترلر (A0) متصل است. در صورتيكه اين پايه 0 باشد، XBee و در صورت 1 بودن كنتور انتخاب ميگردد.
نكته: ارتباط سريال در ميكروكنترلر به صورت data bits:8 ، parity:none و در كنتور به صورت data bits:7 ، parity:even ميباشد. بنابراين براي خوانا بودن دادههاي قرائت شده از كنتور براي ميكروكنترلر، بايد هر بايت داده با 0x7F، AND منطقي شود.
در ادامه برنامه ميكروكنترلر را مشاهده ميكنيد. اين برنامه با نام meter_XBEE در CD همراه گزارش آمده است و با نرمافزار mikroC PRO for PIC v.4.15.0.0 قابل كامپايل شدن است.

برنامه C براي ميكروكنترلر	**توضيح برنامه**
unsigned int i, j; char byte, data; unsigned int Len2Byte;	
Char IeeeByte[8]= {0x00,0x13,0xA2,0x00,0x40,0x3A,0xCC,0xD9}; char LenByte[2]; char buf[30]; char frameID; char checksum; char endflag;	// IEEE Collectorآدرس

char Packet_flag; unsigned long int timeout_count, Packet_timeout_count;	

void main(){
CMCON = 0x07 ; // Disable analog comparators
ADCON1 = 0x06; // Changes PORTA to digital
PORTA = 0;
TRISA = 0;
 	
PORTB = 0;
TRISB = 0;
 	
PORTC = PORTC & 0b11000000;
TRISC = TRISC & 0b11000000;
 	
PORTD = 0;
TRISD = 0;
 	
PORTE = 0;
TRISE = 0;
	پورتهاي A، B، C0 تا C5، D و E خروجي هستند.//

PORTA.F0 = 0;	انتخاب XBee//
Uart1_Init(9600);	سرعت UART براي ارتباط با XBee برابر ۹۶۰۰ ست مي‌شود.//
Packet_flag = 0; Packet_timeout_count = 0; while(1){	
if (Uart1_Data_Ready()){	هنگامي كه ۱ بايت دريافت شد//
data = Uart1_Read(); Packet_timeout_count = 0;	

if (Packet_flag == 0) {
if (data == 0x7E)
Packet_flag = 1;
}
	اگر بايت دريافتي 7E بود يك پكت معتبر دريافت شده است //

else if (Packet_flag == 1) { Packet_flag = 2; LenByte[0] = data;	اگر بايت قبلي 7E بود بايت فعلي را به عنوان بايت پرارزش// طول پكت در نظر مي گيرد//

}
else if (Packet_flag == 2) {
Packet_flag = 3;
LenByte[1] = data;
	اگر بايت اول 7E بود و بايت دوم نيز دريافت شده بود، بايت// سوم را به عنوان بايت كم ارزش طول پكت در نظر مي‌گيرد//

Len2Byte = (LenByte[0]<<8) + LenByte[1];	طول پكت را از روي دو بايت كم ارزش و پرارزش محاسبه// ميكند//

checksum = 0;
i = 0;
}	

else if (Packet_flag == 3) {
 	
if (i< Len2Byte) {
buf[i] = data;
checksum += data;
i++;
}
	اگر بايت اول 7E بود و طول پكت از روي بايت هاي دوم و سوم// محاسبه شده بود، بايتهاي بعدي را تا مقدار طول پكت بافر كن // و مجموع آنها را محاسبه كن//

else{
Packet_flag = 0;
checksum = 0xFF - (checksum & 0xFF);
	بايت دريافتي بايد Checksum باشد، Checksum را از //
	روي مجموع بايتهاي دريافتي محاسبه كن و با اين بايت مقايسه// كن اگر مساوي هم بودند يعني يك پكت معتبر ردياافت شده// است//

if (data == checksum) if ((buf[0]==0x90) && (buf[1]==IeeeByte[0])	حال چك مي كنيم كه اين پكت معتبر

&& (buf[2]==IeeeByte[1]) && (buf[3]==IeeeByte[2]) && (buf[4]==IeeeByte[3]) && (buf[5]==IeeeByte[4]) && (buf[6]==IeeeByte[5]) && (buf[7]==IeeeByte[6]) && (buf[8]==IeeeByte[7]) && (buf[11]==0x01) && (buf[12]==0x00)) {	پکت مورد نظر براي// درخواست قرائت کنتور از جانب collector هست يا نه//
PORTA.F0 = 1;	انتخاب کنتور//
Uart1_Init(300);	سرعت Uart براي ارتباط با کنتور ۳۰۰ ست مي‌شود//
Uart1_Write(0xAF); Uart1_Write(0x3F); Uart1_Write(0x21); Uart1_Write(0x8D); Uart1_Write(0x0A);	ارسال پکت AF 3F 21 8D 0A به کنتور براي Sign on// هر بايت با 0x7F ، AND شده است (پکتي که کنتور// مي‌خواند 2F 3F 21 0D 0A است//

timeout_count = 0;
byte = ' ';
while(byte != 0x0A) {
if (Uart1_Data_Ready()) {
byte = Uart1_Read();
timeout_count = 0;
}
	آنقدر بايت از کنتور قرائت کن تا به اولين 0A برسي//

else {
timeout_count++;
if(timeout_count>100000) {
PORTA.F0 = 0;
Uart1_Init(9600);
break;
}
}
}
	اگر دريافت پاسخ از کنتور Timeout شود از قرائت خارج// شود، XBee را انتخاب کرده و سرعت را به ۹۶۰۰ تغيير دهد//

if (byte == 0x0A) { Uart1_Write(0x06); Uart1_Write(0x30); Uart1_Write(0x35); Uart1_Write(0x30); Uart1_Write(0x8D); Uart1_Write(0x0A);	هنگامي که به اولين 0A رسيدي، در آن صورت پکت // 06 3035 30 8D 0A را به کنتور بفرست//
Delay_ms(300); Uart1_Init(9600);	۳۰۰ ميلي ثانيه صبر کن سپس سرعت را به ۹۶۰۰ تغيير بده//
TRISB = 0; PORTE.F2 = 1; PORTE.F1 = 1; PORTD = 0; PORTC = PORTC & 0b11000000; PORTE.F0 = 0;	تنظيمات لازم براي ارسال بايتهاي خوانده شده به// RAM خارجي//

timeout_count = 0;
byte= ' ';
while(byte != 0x03) {
if (Uart1_Data_Ready()) {
byte = Uart1_Read();
timeout_count = 0;
	آنقدر بايت از کنتور دريافت کن و در RAM بريز تا به// 03 يعني کاراکتر انتهايي برسي//

PORTB = byte & 0x7F; PORTE.F2 = 0; PORTE.F2 = 1;	از آنجايي که در ميکرو databits:8 است بايد هربايت داده// با 7F and شود//

if (PORTD != 0xFF)
PORTD++;
else {
	آدرس را يکي اضافه کن//

PORTD = 0;
if ((PORTC & 0b00111111) != 0b00111111)
PORTC++;
else {
PORTC = PORTC & 0b11000000;
PORTE.F0 = 1;
}
}
}	

else {
timeout_count++;
if(timeout_count>100000){
PORTA.F0 = 0;
Uart1_Init(9600);
break;
}
}
}
	اگر کنتور در حین ارسال اطلاعات Timeout شود از قرائت// خارج شو، XBee را انتخاب و سرعت Uart را ۹۶۰۰ کن//

if (byte == 0x03) {	پس‌ازاینکه داده‌ها در RAM ریخته شد باید از RAM // خوانده شده و به XBee براي ارسال به Collector داده شود//
PORTA.F0 = 0;	انتخاب XBee//
TRISB = 0xFF; PORTE.F2 = 1; PORTD = 0; PORTC = PORTC & 0b11000000; PORTE.F0 = 0; PORTE.F1 = 0;	تنظيمات لازم براي قرائت از RAM خارجي//

frameID = 0x01;
endflag = 1;
while(endflag) {
 	
Uart1_Write(0x7E);
Uart1_Write(0x00);
Uart1_Write(0x56);
Uart1_Write(0x10);
Uart1_Write(frameID);
for(i=0;i<8;i++)
Uart1_Write(IeeeByte[i]);
Uart1_Write(0xFF);
Uart1_Write(0xFE);
Uart1_Write(0x00);
Uart1_Write(0x00);
 	
checksum = 0x0D + frameId;
for(i=0;i<8;i++)
checksum += IeeeByte[i];
	ساخت پکت جهت ارسال به XBee//

for(i=0;i<72;i++){
 	
byte = PORTB;
Uart1_Write(byte);
checksum += byte;
 	
if (PORTD != 0xFF)
PORTD++;
else {
PORTD = 0;
if ((PORTC & 0b00111111) !=
	آدرس را يكي اضافه مي‌كند//

0b00111111)
PORTC++;
else {
PORTC = PORTC & 0b11000000;
PORTE.F0 = 1;
}
}	

if (byte == 0x03)
break;
}
	اگر به کاراکتر انتهايي 03 رسيدي ديگر از RAM نخوان//

if (i<72){
 	
endflag = 0;
for(j=i+1;j<72;j++){
Uart1_Write(0x20);
checksum += 0x20;
}
 	
}
	براي پكت آخر ممكن است ديتا كمتر از ۷۲ بايت يعني طول// پكت باشد. بدين جهت بقيه بايتها را space مي-گذاريم.//

checksum = 0xFF - (checksum & 0xFF); Uart1_Write(checksum);	Checksum را محاسبه كن و به XBee ارسال كن//

timeout_count = 0;
Packet_timeout_count = 0;
while(1){
 	
if (Uart1_Data_Ready()){
data = Uart1_Read();
Packet_timeout_count = 0;
 	
if (Packet_flag == 0) {
if (data == 0x7E)
Packet_flag = 1;
}
 	
else if (Packet_flag == 1) {
Packet_flag = 2;
LenByte[0] = data;
}
 	
else if (Packet_flag == 2) {
Packet_flag = 3;
LenByte[1] = data;
 	
Len2Byte = (LenByte[0]<<8) + LenByte[1];
 	
checksum = 0;
i = 0;
}
 	
else if (Packet_flag == 3) {
 	
if (i< Len2Byte) {
buf[i] = data;
checksum += data;
i++;
}
else{
Packet_flag = 0;
checksum = 0xFF - (checksum & 0xFF);
if (data == checksum)
	منتظر Ack پكت ارسال شده مي‌نشينيم و صحت اين پكت// مانند كد ابتداي برنامه چك مي‌شود//

if ((buf[0]==0x8B) && (buf[1]==frameID))
break;
}
}
}	

else{
Packet_timeout_count++;
if (Packet_timeout_count > 100000) {
Packet_flag = 0;
Packet_timeout_count = 0;
}
}
	اگر پکت در uart، timeout شود ارزیابي پکت را از اول// شروع مي-کند//

timeout_count++;
if(timeout_count>300000)
break;
}
	اگر انتظار براي دريافت Ack، Timeout شود، پکت بعدي را// نمي-فرستيم.//

if ((buf[5] != 0x00) || (timeout_count > 300000)) break;	اگر Ack نشان دهنده شکست ارسال باشد يا انتظار براي دريافت Ack، Timeout شودپکت بعدي را نمي‌فرستيم.

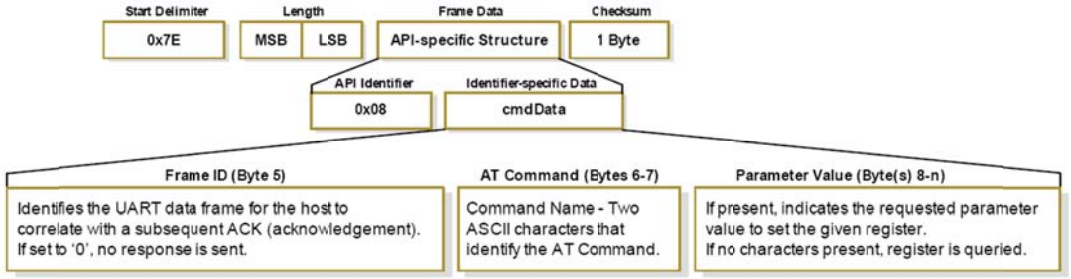
frameID++;
}
}
}
}
}
}
}
	ارسال پکت داده بعدي//

else {
Packet_timeout_count++;
if (Packet_timeout_count > 100000) {
Packet_flag = 0;
Packet_timeout_count = 0;
}
}
}
}
	اگر پکت در uart، Timeout شود، ارزيابي پکت را از اول// شروع مي-کند//

ب-نرم افزار سيستم AMR

اين نرم افزار براي Collector (Coordinator ZigBee) جهت قرائت پريوديک کنتور هاي قرار گرفته در شبکه مش و با استفاده از زبان برنامه نويسي Python نوشته شده است. در اين برنامه، Collector که از طريق پورت RS232 به کامپيوتر متصل است داده هاي قرائت شده از تمامي کنتورهاي شبکه را جمع آوري و در فايلهاي جداگانه ذخيره مي کند. همچنين براي کنتورهاي متصل به بار نمودار انرژي بر حسب زمان را رسم مي کند.
بدين منظور Collector از يک نود درخواست قرائت کنتور مي‌کند. اطلاعات کنتور را دريافت و ذخيره کرده و نمودار انرژي-زمان را براي آن ترسيم مي‌کند. آنگاه سراغ نود بعدي مي‌رود و اين عمل را بدين ترتيب تا آخرين نود تکرار مي‌گردد. اين اعمال طي مراحل زير انجام مي‌پذيرد:
مرحله اول: قبل از ارسال پکت درخواست قرائت، Collector به آدرس IEEE نودها احتياج دارد. از آنجايي که اين آدرس ترتيب مشخصي

ندارد، براي راحتي برنامه هنگام Configuration نودهاي XBee به هر كدام يك NI (Node Identifier) نسبت مي‌دهيم. به اين ترتيب كه به Collector، NI=00 را نسبت داده و به بقيه نودها به ترتيب01، 02، 03،...، 09، 0A، 0B،...،0F، 10، 11، 12،... را نسبت مي‌دهيم (لازم به ذكر است كه هر چند اين مقادير عدد هستند اما در Node Identifier در نرم افزار X-CTU به عنوان كاراكتر در نظر گرفته مي‌شوند). بدين ترتيب در اولين مرحله از برنامه Collector براي كشف آدرس IEEE نود، دستور Node Discovery (AT ND [NI Number]) را به نود مربوطه مي‌فرستد. ساختار اين دستور مطابق با ساختار پكت AT Command است كه در شكل ۲-۱۷ نشان داده شده است.

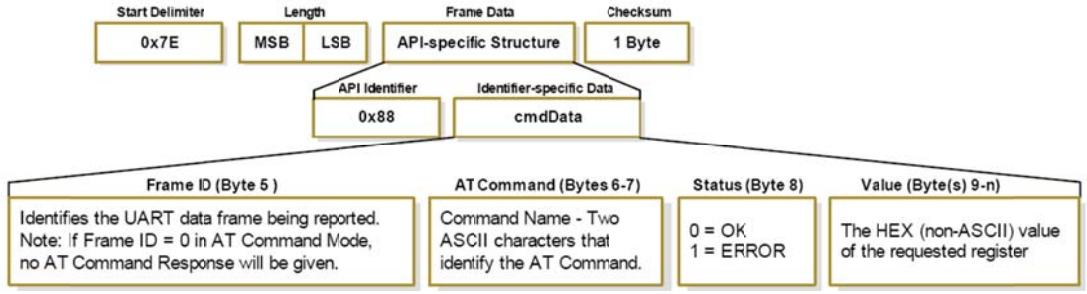


شكل۲-۱۷-ساختار پكت AT Command

بر اين اساس پكت ارسالي به نود NI=01 به صورت زير است:

7E	00	06	08	01	4E	44	30	31	03
					N	D	0	1	

مرحله دوم: پس از ارسال اين پكت، Collector پاسخي از نود مربوطه بر اساس ساختار پكت AT Command Response (مطابق شكل۲-۱۸) دريافت مي‌كند كه در آن آدرس IEEE نود مربوطه قرار دارد.



شكل۲-۱۸-ساختار پكت AT Command Response

بر اين اساس پاسخ دريافتي از نود مربوطه به عنوان مثال نودي با آدرس 00 13 A2 00 40 3A CC D7 به صورت زير است:

7E 00 1A 88 01 4E 44 00 FF FE	00 13 A2 00 40 3A CC D7	30 31	00 FF FE 01 00 C1 05 10 1E C2
	IEEE Address	NI:01	

مرحله سوم: از پكت پاسخ دريافتي، بايت‌هاي آدرس را جدا و ذخيره مي‌كند.

مرحله چهارم: بر اساس ساختار پكت Transmit Request (شكل ۲-۱۵) پكت درخواست قرائت كنتور (پكتي كه بخش داده در آن يك بايت 00

است) را به نودي كه آدرس آن را در مرحله قبل دريافت كرده بود ارسال مي‌كند. اين پكت براي ارسال به نودي با آدرس 00 13 A2 00 40 3A CC D7 به صورت زير است:

7E 00 0F 10 00	00 13 A2 00 40 3A CC D7	FF FE 00 00 00 20
	IEEE Address	

مرحله پنجم: Collector، پاسخ نود مربوطه را بر اساس ساختار Receive Packet (شكل۲-۱۴) دريافت كرده، اطلاعات مربوط به كنتور را از پكت دريافتي استخراج كرده و در فايل مربوط به آن كنتور ذخيره مي‌كند. پكت دريافتي با فرض ۷۲ بايت داده صفر به صورت زير خواهد بود.

7E	00	54	90	00	13	A2	00	40	3A	CC	D7	FF	FE	01	00	...	00	9F
				IEEE Address فرستنده											با فرض ۷۲ بايت داده‌ي صفر			

مرحله ششم: مراحل بالا تكرار مي‌شود تا اطلاعات كليه كنتورها ذخيره گردد.

مرحله هفتم: نمودارهاي مصرف براي كنتورها ترسيم مي‌گردد.

براي اجراي برنامه Coordinator، نياز به نصب Python2.5 مي‌باشد كه مي‌توان آن را از http://search.4shared.com/q/CCAD/1/python+2.5 دانلود كرد. به علاوه براي برقراري ارتباط سريال نياز به نصب نرم افزار pyserial مي‌باشد كه در CD گزارش قرار داده شده است. علاوه بر اين، براي ذخيره اطلاعات كنتورها لازم است در جايي كه اين برنامه ذخيره شده است، يك فولدر به نام ReadRegisters ساخته شده و به تعداد كنتورها فايل‌هايtext با نام‌هاي Counter1، Counter2 و... ساخته شود. اين برنامه با نام WinPlotCollector ND در CD همراه گزارش آمده است.

برنامه AMR براي Collector به زبان Python

```
print '\n***************************************************'
print '       AUTOMATIC METER READER PROGRAM          '
print '                 IS RUNNING                    '
print '***************************************************\n'


import os
from pylab import *
from msvcrt import *

import serial
import time
#setting of serial port
ser=serial.Serial()

if ser.isOpen():
      ser.close()

ser.port = 'COM1:'
ser.baudrate = 9600
ser.bytesize = serial.EIGHTBITS
ser.parity = serial.PARITY_NONE
ser.stopbits = serial.STOPBITS_ONE
ser.timeout = 1

ser.open()
```

```
LoopNum=1     #first entry in data collection loop
maxLoops=3   #number of times data collection is occured (each 15 seconds)

print ser

MaxCounterNum = 1                #تعداد کنتورها

out_file=[]
for i in range(0,MaxCounterNum) :
      out_file.append([])

NI = []
for i in range(0,2) :
      NI.append([])

IeeeByte = []
for i in range(0,8) :
      IeeeByte.append([])

LenByte=[]
for i in range(0,2) :
      LenByte.append([])

Hex_Symbol = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F']

NIByte = 1              #از نود شماره ۱ شروع مي كند

Counter_timeout = 15        #زمان بر حسب ثانيه
Packet_timeout = 0.5

# Initial positions
x = []
y = []
h = []
time_tick = []

fileDateYear_pre = []
fileDateMonth_pre = []
fileDateDay_pre = []

fileTimeHours_pre = []
fileTimeMinutes_pre = []
fileTimeSeconds_pre = []

for CounterCode in range(0,MaxCounterNum) :
   h.append([])
   x.append([])
   y.append([])
   time_tick.append([])

   fileDateYear_pre.append(99)  مثلاً ۹۹ عددي است كه هيچوقت سال، ماه، روز، ساعت، دقيقه و ثانيه برابر آن
     نمي شود#
   fileDateMonth_pre.append(99)
   fileDateDay_pre.append(99)

   fileTimeHours_pre.append(99)
   fileTimeMinutes_pre.append(99)
   fileTimeSeconds_pre.append(99)

while (LoopNum<=maxLoops) :

      ser.flushInput()
      ser.flushOutput()

      print '\n----------------------------------------------'
      print '\nDiscovering Node',
      print NIByte,
      print 'IEEE Address...',

      NI[0] = ord( Hex_Symbol[((NIByte >> 4) & 0x0F)] )     #NI MSB
      NI[1] = ord( Hex_Symbol[NIByte & 0x0F] )              #NI LSB
                                                  ### ATND NI[0]NI[1]  ارسال اين دستور براي دريافت آدرس ۱۶
بيتي نود
      checksum = 0xFF - ( 0x9B + NI[0] + NI[1] ) & 0xFF
      ser.write( chr(0x7E) + chr(0x00) + chr(0x06) + chr(0x08) + chr(0x01) + chr(0x4E) + chr(0x44) + chr(NI[0]) + chr(NI[1]) +
```

```
chr(checksum) )

        timeout_flag = 1
        Packet_init_time = Counter_init_time = time.time()

        Packet_flag = 0

        while timeout_flag :

                if ( time.time()-Counter_init_time ) >          #اگر پاسخ نود Timeout شود سراغ نود بعدي مي رود
Counter_timeout :
                        timeout_flag = 0
                        print 'Timeout'
                        break

                if ser.inWaiting() > 0 :
                        data = ser.read(1)

                        Packet_init_time = time.time()

                        if Packet_flag == 0  :
                                if data == chr(0x7E) :
                                        Packet_flag = 1

                        elif Packet_flag == 1 :
                                Packet_flag = 2
                                LenByte[0] = ord(data)

                        elif Packet_flag == 2 :
                                Packet_flag = 3
                                LenByte[1] = ord(data)

                                Len2Byte = ( LenByte[0]<<8 ) + LenByte[1]

                                buf = ''
                                checksum = 0
                                i=0

                        elif Packet_flag == 3 :

                                if i < Len2Byte :
                                        buf += data
                                        checksum += ord(data)
                                        i += 1
                                else :
                                        Packet_flag = 0
                                        checksum = 0xFF - (checksum & 0xFF)
                                        if data == chr(checksum) :
                                                try :
                                                        if ( buf[0] == chr(0x88) ) & ( buf[1] == chr(0x01) ) & ( buf[2] == chr(0x4E) ) & ( buf[3] == chr(0x44) )
& ( buf[4] == chr(0x00) ) & ( buf[15] == chr(NI[0]) ) & ( buf[16] == chr(NI[1]) ):              #اگر پکت معتبر تشخيص داده شود

                                                                Counter_init_time = time.time()
                                                                                                #آدرس نود استخراج مي شود
                                                                IeeeByte[0] = ord(buf[7])
                                                                IeeeByte[1] = ord(buf[8])
                                                                IeeeByte[2] = ord(buf[9])
                                                                IeeeByte[3] = ord(buf[10])
                                                                IeeeByte[4] = ord(buf[11])
                                                                IeeeByte[5] = ord(buf[12])
                                                                IeeeByte[6] = ord(buf[13])
                                                                IeeeByte[7] = ord(buf[14])

                                                                print 'Discovered'

                                                                print '\nRequesting counter',
                                                                print NIByte,
                                                                print ' registers...',
#پکت درخواست قرائت کنتور به نود مربوطه ارسال مي شود

                                                                checksum = 0xFF - ( 0x0D + sum(IeeeByte) ) & 0xFF
                                                                ser.write( chr(0x7E) + chr(0x00) + chr(0x0F) + chr(0x10) + chr(0x00) + chr(IeeeByte[0]) +
chr(IeeeByte[1]) + chr(IeeeByte[2]) + chr(IeeeByte[3]) + chr(IeeeByte[4]) + chr(IeeeByte[5]) + chr(IeeeByte[6]) + chr(IeeeByte[7]) +
chr(0xFF) + chr(0xFE) + chr(0x00) + chr(0x00) + chr(0x00) + chr(checksum) )
```

```
print 'Requested'

timeout_flag = 1
Packet_init_time = time.time()

print '\nWaiting for counter',                    #منتظر پاسخ کنتور مي نشيند
print NIByte,
print ' response...',

Packet_flag = 0

while timeout_flag :

#اگر پاسخ timeout شود فايل را بسته و سرغ نود بعدي ميرود
    if ( time.time()Counter_init_time ) > Counter_timeout :
        timeout_flag = 0
        print 'Timeout'
        try :
            out_file[NIByte-1].close()
            print '\nFile closed'

        except ( NameError, ValueError, AttributeError ) :
            pass
        break
                                        #اگر پکت معتبر تشخيص داده شود
    if ser.inWaiting() > 0 :
        data = ser.read(1)

        Packet_init_time = time.time()

        if Packet_flag == 0  :
            if data == chr(0x7E) :
                Packet_flag = 1

        elif Packet_flag == 1 :
            Packet_flag = 2
            LenByte[0] = ord(data)

        elif Packet_flag == 2 :
            Packet_flag = 3
            LenByte[1] = ord(data)

            Len2Byte = ( LenByte[0]<<8 ) + LenByte[1]

            buf = ''
            checksum = 0
            i=0

        elif Packet_flag == 3 :

            if i < Len2Byte :
                buf += data
                checksum += ord(data)
                i += 1
            else :
                Packet_flag = 0
                checksum = 0xFF - (checksum & 0xFF)
                if data == chr(checksum) :# Check the recieved packet from countor node#####
                    try :          #اگر پکت معتبر پاسخ نود مورد نظر باشد
                        if ( buf[0] == chr(0x90) ) & ( buf[1] == chr(IeeeByte[0]) ) & ( buf[2] == chr(IeeeByte[1]) ) & ( buf[3] == chr(IeeeByte[2]) ) & ( buf[4] == chr(IeeeByte[3]) ) & ( buf[5] == chr(IeeeByte[4]) ) & ( buf[6] == chr(IeeeByte[5]) ) & ( buf[7] == chr(IeeeByte[6]) ) & ( buf[8] == chr(IeeeByte[7]) ) & ( buf[11] == chr(0x01) ) :

                            Counter_init_time = time.time()

                            print
                            print  buf[12:Len2Byte] #داده هاي كنتور را چاپ مي كند

                            #داده ها را در فايل مربوط مي نويسد

                            print '\nWriting to file...',
                            try :
                                out_file[NIByte-1].write( buf[12:Len2Byte] )
                            #### make sure to make the folder and the text files in it ######
```

```
                                                    except ( NameError, ValueError, AttributeError ) :
                                                        out_file[NIByte-1] = open( 'ReadRegisters/Counter' +
str(NIByte) + '.txt', 'w' )
                                                        out_file[NIByte-1].write( buf[12:Len2Byte] )

                                                    print 'Writed'
اگر کاراکتر انتهايي يعني ۳. در ديتاي پکت وجود داشت يعني آخرين پکت از داده هاي کنتور است پس فايل مربوطه را مي بندد#
                                                    if buf[12:Len2Byte].find( chr(0x03) ) != -1 :
                                                        out_file[NIByte-1].close()
                                                        print '\nFile closed'
                                                        timeout_flag = 0
                                                        break

                                                except IndexError :
                                                    pass
                                    اگر پکتtimeout شود ارزيابي پکت را از اول شروع مي کند#
                                                elif ( time.time()-Packet_init_time ) > Packet_timeout :
                                                    Packet_flag = 0
                                                    Packet_init_time = time.time()

                                    except IndexError :
                                        pass
                        اگر پکتtimeout شود ارزيابي پکت را از اول شروع مي کند#

                        elif ( time.time()-Packet_init_time ) > Packet_timeout :
                            Packet_flag = 0
                            Packet_init_time = time.time()

                    سراغ نود بعدي مي رود#
                NIByte += 1
                #print timeout_flag
            وقتي همه کنتورها قرائت شدند نمودار active energy را بر حسب time-date براي هر کنتور ترسيم مي کند#
            if (NIByte > MaxCounterNum) :
                print '\n**************************************************'
                print '       All Counter registers requested'
                print '**************************************************'

                for CounterCode in range(0,MaxCounterNum) :

                    print '\nPlotting active energy vs time-date for counter',
                    print CounterCode+1,
                    print '...',

                    SerialNumber_flag = 0
                    Time_flag = 0
                    Date_flag = 0
                    Energy_flag = 0

                    try :                   فايل مربوط به کنتور را باز مي کند#

                        in_file = open('D:/Zihajehzadeh/Python/codes/ReadRegisters/Counter' + str(CounterCode+1) + '.txt','r')

                        # Extract Time, Date and Energy
                        for fileLine in in_file.readlines():

                            # Read serial Number
                            if fileLine.find('0.0.0(') != -1 :
                                fileSerialNumber=fileLine.split('0.0.0(')[1].split(')')[0]
                                SerialNumber_flag = 1

                            # Read time
                            elif fileLine.find('0.9.1(') != -1 :
                                fileTime=fileLine.split('0.9.1(')[1].split(')')[0]
                                try :
                                    fileTimeHours= int( fileTime[0:2] )
                                    fileTimeMinutes= int( fileTime[2:4] )
                                    fileTimeSeconds= int( fileTime[4:6] )
                                    Time_flag = 1
                                except ValueError:
                                    pass

                            # Read date
                            elif fileLine.find('0.9.2(') != -1 :
                                fileDate=fileLine.split('0.9.2(')[1].split(')')[0]
```

```
                        try :
                            fileDateYear= int( fileDate[0:2] )
                            fileDateMonth= int( fileDate[2:4] )
                            fileDateDay= int( fileDate[4:6] )
                            Date_flag = 1
                        except ValueError:
                            pass

                    # Read energy
                    elif fileLine.find('1.8.0(') != -1 :
                        try :
                            fileEnergy=float( fileLine.split('1.8.0(')[1].split('*kWh)')[0] )
                            Energy_flag = 1
                        except ValueError:
                            pass

                    elif fileLine.find('128.8.0(') != -1 :
                        try :
                            fileEnergy=float( fileLine.split('128.8.0(')[1].split('*kWh)')[0] )
                            Energy_flag = 1
                        except ValueError:
                            pass

                    if SerialNumber_flag and Time_flag and Date_flag and Energy_flag :
                        break

                # Close file
                in_file.close()

                if SerialNumber_flag and Time_flag and Date_flag and Energy_flag :

                    time_string = str(fileTimeHours)+':'+str(fileTimeMinutes)+':'+str(fileTimeSeconds)
                    date_string = str(fileDateYear)+'/'+str(fileDateMonth)+'/'+str(fileDateDay)

                    if not ( ( fileDateYear == fileDateYear_pre[CounterCode] ) and ( fileDateMonth ==
fileDateMonth_pre[CounterCode] ) and ( fileDateDay == fileDateDay_pre[CounterCode] ) ) :

                        # Plot
نمودار به صورت روزانه رسم مي شود پس اگر روز يا ماه يا سال تاريخ قرائت شده با قرائت قبلي فرق داشته باشد نمودار جديد مي كشد#
                        x[CounterCode] = [fileTimeHours*60*60 + fileTimeMinutes*60 + fileTimeSeconds]
                        y[CounterCode] = [fileEnergy]

                        time_tick[CounterCode] = [time_string]
                        مشخصات شكل را تعيين مي كند و شكل را به صورت يك فايل PNG ذخيره مي كند#
                        figure( figsize=(15, 10) )
                        plot( x[CounterCode], y[CounterCode], '-o', label = 'Date: ' + date_string )
                        grid()
                        xlabel('Time')
                        ylabel('Active Energy (kWh)')
                        title( 'Active Energy vs Time\nSerial Number: ' + fileSerialNumber )
                        legend( loc='best' )
                        xticks( x[CounterCode], time_tick[CounterCode], rotation='vertical', size='x-small' )
                        savefig( 'Plots/Day'+str(fileDateDay)+'/Counter'+str(CounterCode+1)+'.png' )
                        close()

                        fileDateYear_pre[CounterCode] = fileDateYear
                        fileDateMonth_pre[CounterCode] = fileDateMonth
                        fileDateDay_pre[CounterCode] = fileDateDay

                    elif not ( ( fileTimeHours == fileTimeHours_pre[CounterCode] ) and ( fileTimeMinutes ==
fileTimeMinutes_pre[CounterCode] ) and ( fileTimeSeconds == fileTimeSeconds_pre[CounterCode] ) ) :

                        # Plot
اگر قرائت مربوط به همان روز باشد اما در يك لحظه متفاوت انگاه نقطه قرائت جديد را به نمودار قبلي اضافه كرده و ذخيره مي كند#
                        x[CounterCode].append( fileTimeHours*60*60 + fileTimeMinutes*60 + fileTimeSeconds )
                        y[CounterCode].append(fileEnergy)

                        time_tick[CounterCode].append( time_string )
                                    مشخصات شكل را تعيين مي كند#
                        figure( figsize=(15, 10) )
                        plot( x[CounterCode], y[CounterCode], ‘-o’, label = ‘Date: ‘ + date_string )
                        grid()
                        xlabel(‘Time’)
```

```
                        ylabel('Active Energy (kWh)')
                        title( 'Active Energy vs Time\nSerial Number: ' + fileSerialNumber )
                        legend( loc='best' )
                        xlim( x[CounterCode][0], x[CounterCode][-1] )
                        ylim( y[CounterCode][0], y[CounterCode][-1] )
                        xticks( x[CounterCode], time_tick[CounterCode], rotation='vertical', size='x-small' )
                        savefig( 'Plots/Day'+str(fileDateDay)+'/Counter'+str(CounterCode+1)+'.png' )
                        close()


                    fileTimeHours_pre[CounterCode] = fileTimeHours
                    fileTimeMinutes_pre[CounterCode] = fileTimeMinutes
                    fileTimeSeconds_pre[CounterCode] = fileTimeSeconds

                print 'Plotted'

            except IOError :
                print 'No Data'

        print '\n**************************************************'
        print '      All Counters active energy vs time-date Plotted'
        print '**************************************************'
                #۱۰ دقیقه صبر می کند و مجدداً سراغ اولین نود می رود
        NIByte = 1
        LoopNum=LoopNum+1
        for i in range(0,1):
            time.sleep(15)
print
print 'Exiting The Program.....'
print

ser.close()
close('all')
```

لازم به ذکر است که برای آپلود کردن فایل کنتورها روی یک ftp لازم است از ftplib در Python استفاده شود. برنامه آپلود کردن تحت عنوان WinPlotCollector_ND_ftp در CD همراه گزارش قرار داده شده است. در این برنامه تنها چند خط کد برای آپلود کردن نسبت به برنامه WinPlotCollector_ND اضافه شده است.

۲-۳-۳-نتایج پیاده‌سازی پایلوت

برای پیاده‌سازی سیستم AMR پیشنهادی مراحل زیر باید انجام شود:

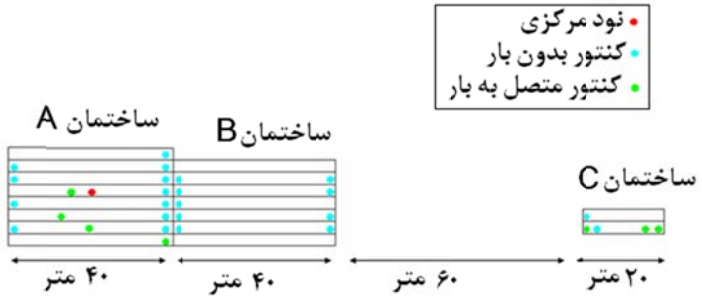
۱- به تعداد کنتورهای موجود در شبکه مدار الکترونیکی بخش فرستنده (مدار شکل۲-۱۰) ساخته شود و میکروکنترلر روی آن برنامه‌ریزی گردد. XBee های روی این بردها باید به صورت Router برنامه‌ریزی شده باشند. به علاوه شماره NI این XBeeها باید به ترتیب ، 02، 03،...، 09، 0A، 0B،...،0F، 10، 11، 12،... ست شود.

۲- به هر کنتور یک مدار الکترونیکی فرستنده متصل گردد (از طریق کابل پورت نوری که از یک طرف به کنتور و از طرف دیگر به مدار فرستنده متصل است.)

۳- یک XBee به صورت Coordinator برنامه ریزی گردد و از طریق Starter Kit به کامپیوتری که برنامه Python برای Collector قرار است اجرا گردد متصل شود. این XBee به عنوان Collector مورد استفاده قرار گرفته و شماره NI آن باید 00 ست شود.

۴- برنامه Python بر روی پردازنده‌ای که Coordinator به آن متصل است اجرا گردد و اطلاعات کنتورها جمع‌آوری شود.

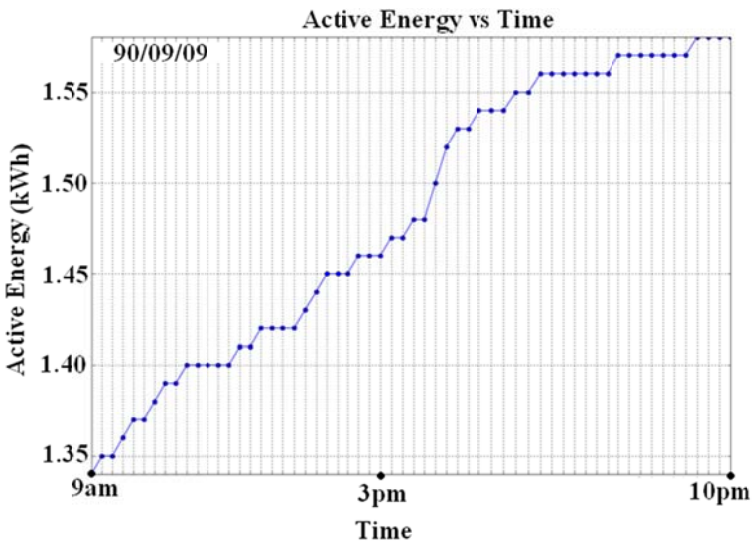
به عنوان یک پروژه پایلوت، سیستم AMR پیشنهادي براي چند ساختمان در شهر اصفهان پیاده‌سازي شده است. در این پروژه آزمایشی تعداد چهل کنتور برق در طبقات مختلف ساختمان شرکت فولادتکنیك، قائم رضا و نیز ساختمان آرشیو قرار داده شده‌اند. تعدادی از این کنتورها جهت ترسیم نمودار مصرف انرژی به بار مصرفی متصل شده اند. موقعیت و فواصل تقریبی این کنتورها در شکل۲-۱۹ نشان داده شده است.



شکل ۲-۱۹- موقعیت کنتورها

این کنتورها داده های قرائت شده خود را از طریق یک شبکه مش ZigBee یک نود مرکزی ، ارسال می نمایند. نود مرکزی داده های قرائت شده از تمامی کنتورهای شبکه را جمع آوری و ذخیره می کند و برای کنتورهای متصل به بار نمودار مصرف انرژی را ترسیم می نماید و پس از جمع‌آوري کلیه اطلاعات، با استفاده از مودم GPRS آن‌ها را بر روي سرور مورد نظر قرار مي‌دهد. شکل ۲-۲۰ نمودار مصرف انرژی بر حسب زمان را به صورت نمونه برای یك بار مصرفی (پرینتر) نشان می دهد.

براي آزمودن قابلیت اطمینان این شبکه تعدادي از نود هاي میاني از شبکه خارج گردیده و بار دیگر نمودار مصرف انرژي بر حسب زمان براي بارهاي مذکور ترسیم شده است. این آزمایش نشان داد که مطابق انتظار، در صورت قطع یک یا چند نود در این شبکه، ارتباط بقیه نودها با نود مرکزی قطع نشده و آنها می توانند از طریق مسیرهای دیگر ارتباط خود را با نود مرکزی حفظ نمایند. به دلیل وجود این مسیرهای چندگانه قابلیت اطمینان شبکه بسیار بالا و در حد شبکه‌هاي با سیم است.



شکل ۲-۲۰- نمودار مصرف انرژی بر حسب زمان پرینتر