**Software Architecture to Integrate Sensors and Controllers**

A system architecture is proposed for integrating sensors, controllers, actuators and instrumentation within a common framework. The goal is to provide a flexible and scalable system. Extending the system, by adding additional components such as sensors or actuators, does not increase the overheads and is achieved seamlessly with minimal modification of the core controller program. The architecture is generic, though it has been proposed to implement a prototype system on a small form factor PC to remote control an autonomous vehicle. The architecture will find application in many other spheres such as home, office and factory automation, process and environmental monitoring, surveillance and robotics.

معماری سیستم برای ادغام سنسورها ، کنترلرها ، محرکها و ابزار دقیق در یک چارچوب مشترک پیشنهاد شده است. هدف این است که یک سیستم انعطاف پذیر و مقیاس پذیر ارائه شود. گسترش سیستم ، با اضافه کردن مؤلفه های اضافی مانند سنسورها یا محرک ها ، باعث افزایش سربار نمی شود و با کمترین تغییر در برنامه کنترل هسته یکپارچه حاصل می شود. معماری عمومی است ، اگرچه پیشنهاد شده است برای پیاده سازی یک سیستم نمونه اولیه بر روی رایانه شخصی با فرم کوچک برای کنترل از راه دور یک وسیله نقلیه خودمختار پیشنهاد شود. این معماری در بسیاری از زمینه های دیگر مانند اتوماسیون خانگی ، اداری و کارخانه ای ، نظارت بر فرآیند و محیط زیست ، نظارت و رباتیک کاربردهایی پیدا خواهد کرد.

**1 Introduction**

Sensors are used in a variety of environments, with varying platforms. In the real world, sensory data comes from multiple sensors of different modalities in either a distributed or centralised location. There are enormous challenges in collecting data, evaluating the information, performing decision-making, formulating meaningful user displays and actuating alarms or other controls.

1. مقدمه

سنسورها در محیط های مختلف و دارای سیستم عامل های مختلف استفاده می شوند. در دنیای واقعی ، داده های حسی از چندین حسگر از روش های مختلف در یک مکان توزیع شده یا متمرکز تهیه می شود. در جمع آوری داده ها ، ارزیابی اطلاعات ، انجام تصمیم گیری ، تدوین نمایشنودای معنی دار کاربر و فعال کردن هشدارها یا سایر کنترل ها ، چالش های بزرگی وجود دارد.

In the last decade, tremendous research efforts have been concentrated in the area of intelligent networked sensors and wireless network sensors addressing issues of self-organization, scalability and data flow management. A system architecture for monitoring habitat has been proposed. A tiered architecture comprising sensor nodes, connected to a base station via transit networks, Internet connectivity to data services, and user interface for data display has been developed. Their architecture does not incorporate any actuators. Moreover, the emphasis is on network architecture, rather than the controller architecture.

در دهه گذشته ، تلاشهای پژوهشی عظیمی در زمینه حسنودای شبکه هوشمند و سنسورهای شبکه بی سیم صورت گرفته است که به موضوعات خود ساماندهی ، مقیاس پذیری و مدیریت جریان داده می پردازند. یک معماری سیستم برای نظارت بر زیستگاه ارائه شده است. یک معماری لایه دار شامل نود های حسگر ، متصل به یک ایستگاه پایه از طریق شبکه های ترانزیت ، اتصال اینترنت به خدمات داده و رابط کاربری برای نمایش داده ها توسعه داده شده است. معماری آنها هیچ محرکی را شامل نمی شود. علاوه بر این ، تأکید بیشتر بر معماری شبکه است نه معماری کنترلر.

presents protocol architectures for scalable self-assembly of sensor networks. makes a case for a data-centric network in which data is named by attributes. This approach decouples data from the sensor that produces it allowing for more robust application design. The CODGER system [4], developed for large remote vehicles, addresses the overall architecture of a sensor system. The manager keeps a list of which tasks are ready to execute, and in its central loop selects one task to execute. Each task has a pre-determined pattern of data that must be satisfied before it can be considered for execution. Tasks have access to data at various levels and each task is a separate, continuously running program.

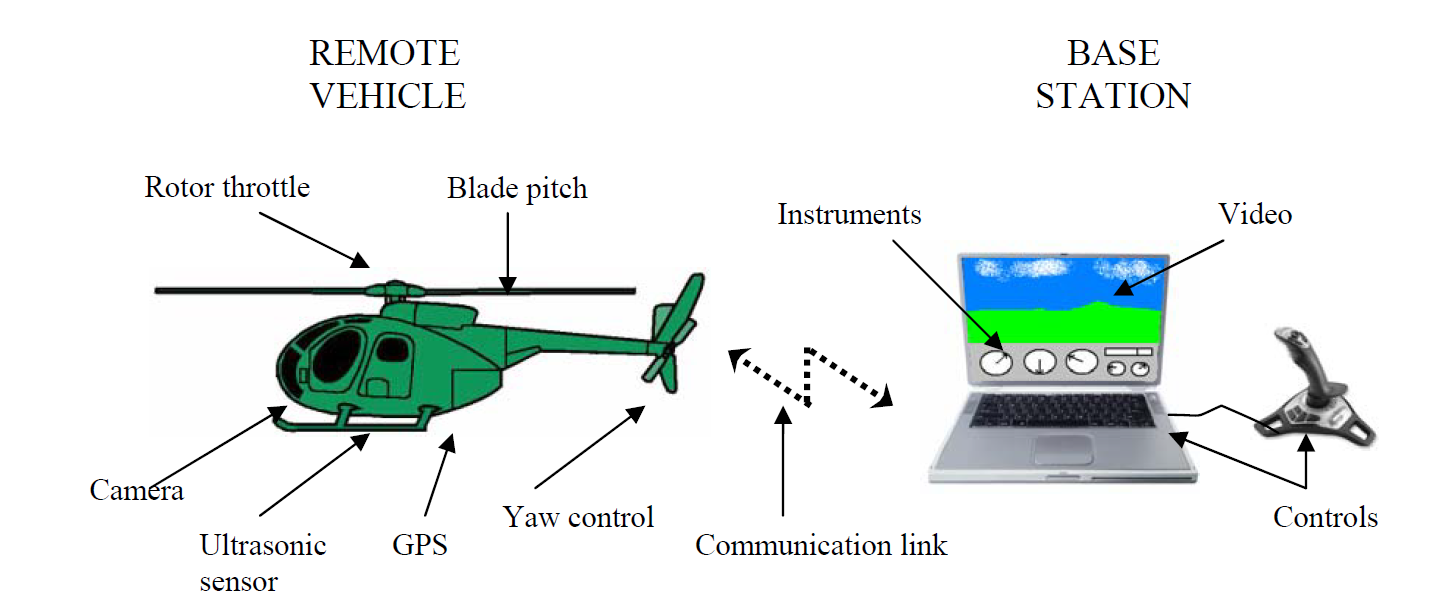
ارائه معماری پروتکل برای مونتاژ مقیاس پذیر شبکه های حسگر. یک شبکه داده محور را ایجاد می کند که در آن داده ها با ویژگی ها نامگذاری می شوند. این روش داده های سنسور را مجزا می کند و باعث می شود طراحی برنامه ای قوی تر انجام شود. سیستم CODGER ، که برای وسایل نقلیه بزرگ از راه دور ساخته شده است ، به معماری کلی سیستم حسگر می پردازد. مدیر لیستی از وظایف آماده انجام را نگه می دارد و در حلقه مرکزی خود یک کار را برای اجرا انتخاب می کند. هر کار الگویی از پیش تعیین شده ای از داده ها دارد که قبل از در نظر گرفتن آن برای اجرا باید انجام شود. کارها در سطوح مختلف به داده دسترسی دارند و هر کار یک برنامه جداگانه و مداوم است.

While there is enough work that has been done on sensor fusion, no evidence is available that leads us to believe that attempts have been made to design a flexible architecture to integrate sensors, actuators, controller and instrumentation within a single framework. In a scalable system, it should be easy to add components, such as sensors and actuators, seamlessly as this happens frequently. In most systems, this would entail considerable system reconfiguration, resource re-mapping and possibly change of control methods. This has motivated the authors to undertake this work and we believe that a generic integrated sensor and controller architecture would be beneficial in many applications.

در حالی که کار کافی در مورد فیوژن سنسور انجام شده است ، هیچ مدرک در دسترس نیست که ما را به این باور برساند که تلاش شده است برای طراحی یک معماری انعطاف پذیر برای یکپارچه سازی سنسورها ، محرک ها ، کنترل کننده و ابزار دقیق در یک چهارچوب واحد انجام شود. در یک سیستم مقیاس پذیر ، می توان اجزاء مانند سنسورها و محرک ها را به آسانی اضافه کرد زیرا این اتفاق اغلب رخ می دهد. در اکثر سیستم ها ، این امر مستلزم پیکربندی مجدد سیستم ، نقشه برداری مجدد منابع و احتمالاً تغییر روش های کنترل است. این امر نویسندگان را به انجام این کار انگیزه داده است و ما معتقدیم که یک سنسور یکپارچه عمومی و معماری کنترل کننده در بسیاری از برنامه ها سودمند خواهد بود.

The proposed architecture uses a data driven approach. The data is passed from one task (module) to another, with each module processing the data in some way. Each processing module is independent. There is separate processing for data acquisition, communication and actuation. This makes modifying or adding further processing blocks, for example to add new features when required, quite straightforward.

معماری پیشنهادی از یک رویکرد داده محور استفاده می کند. داده ها از یک کار (ماژول) به کار دیگر منتقل می شوند که هر ماژول داده ها را به نوعی پردازش می کند. هر ماژول پردازش مستقل است. پردازش جداگانه ای برای کسب اطلاعات ، ارتباطات و فعال سازی داده ها وجود دارد. این امر باعث می شود تا بلوک های پردازش دیگری اصلاح یا اضافه شود ، به عنوان مثال برای افزودن ویژگی های جدید در مواقع لزوم ، کاملاً ساده.



شکل 1: تنظیمات سیستم معمولی برای کنترل وسیله نقلیه از راه دور که نشان دهنده استفاده از سنسورها ، ابزارها ، کنترل ها و محرک ها است

In this paper we present the details of the integrated sensor and controller architecture using the control of a remote vehicle from a base station as an example. This configuration is illustrated in Figure 1 where a remote model helicopter is controlled for surveillance purposes. However, since this architecture is flexible, scalable and extensible it could be used for a variety of other applications such as home, office and factory automation, process and environmental monitoring, surveillance and robotics.

در این مقاله جزئیات سنسور یکپارچه و معماری کنترلر با استفاده از کنترل یک وسیله نقلیه از راه دور از یک ایستگاه پایه به عنوان نمونه ارائه می کنیم. این پیکربندی در شکل 1 نشان داده شده است که یک هلیکوپتر مدل از راه دور برای اهداف نظارت کنترل می شود. اما از آنجا که این معماری انعطاف پذیر ، مقیاس پذیر و قابل توسعه است ، می توان از آن برای انواع کاربردهای دیگر مانند اتوماسیون خانگی ، اداری و کارخانه ، فرآیند و نظارت بر محیط زیست ، نظارت و روباتیک استفاده کرد.

**2 System Architecture**

To simplify the design, a common architecture is used for both the remote vehicle and the base station. The architecture has a number of desired features. To be general purpose, it must be able to take readings from a variety of sensors. These sensors can be on different buses with varying protocols. It may need to perform custom processing or manipulation of the data obtained. This includes converting or stripping the raw information from the sensor to get information that can be used. Another key function of the architecture is to facilitate the timely communication of sensor and control data between the remote and base station.

1. معماری سامانه

برای ساده سازی طراحی ، معماری مشترکی هم برای وسیله نقلیه از راه دور و هم برای ایستگاه پایه استفاده می شود. این معماری دارای چندین ویژگی مورد نظر است. برای اینکه هدف کلی باشد ، باید بتواند از سنسورهای مختلفی قرائت کند. این سنسورها می توانند در اتوبوس های مختلف با پروتکل های مختلف باشند. ممکن است نیاز به انجام پردازش های سفارشی یا دستکاری داده های به دست آمده باشد. این شامل تبدیل یا سلب اطلاعات خام از سنسور برای به دست آوردن اطلاعاتی است که می تواند مورد استفاده قرار گیرد. یکی دیگر از کارکردهای مهم معماری تسهیل ارتباط به موقع سنسور و کنترل داده ها بین ایستگاه از راه دور و پایه است.

The communication channel could be wired or wireless and based on a number of different protocols. The display shows processed data on virtual instruments. For remote vehicle control, the base station may be configured as a virtual cockpit. The architecture accepts control input from the base station. It then communicates this control information to the remote platform. The proposed architecture also allows for easy integration of autonomous control operations.

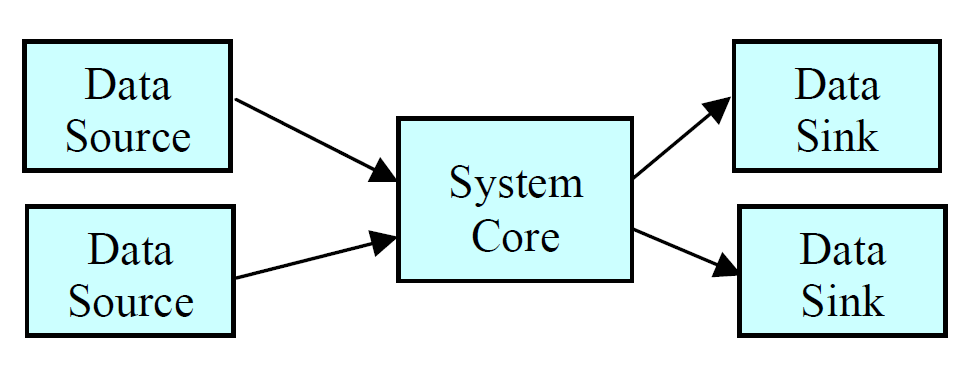
کانال ارتباطی می تواند بی سیم یا بی سیم باشد و بر اساس تعدادی پروتکل مختلف ساخته شود. صفحه نمایش داده های پردازش شده روی ابزارهای مجازی را نشان می دهد. برای کنترل وسیله نقلیه از راه دور ، ایستگاه پایه ممکن است به عنوان یک کابین خلبان مجازی تنظیم شود. معماری ورودی کنترل را از ایستگاه پایه می پذیرد. سپس این اطلاعات کنترل را به بستر راه دور ارتباط می دهد. معماری پیشنهادی همچنین امکان ادغام آسان عملیات کنترل مستقل را فراهم می آورد.

These requirements suggest that a data-driven approach is most appropriate for the system architecture. In this context, it is observed that every device may be thought of as a data source or data sink as shown in the basic system structure in Figure 2. Sensors and controls are both data sources. Actuators and virtual instruments are data sinks. Data conversion and autonomous operations are both sinks and sources, with the sink corresponding to the input to the function and the processed result or control output being a data source. Each direction of the communications channel may be considered a data sink at the transmitting end, and a data source at the receiver.

این الزامات نشان می دهد که یک رویکرد مبتنی بر داده برای معماری سیستم مناسب ترین است. در این زمینه مشاهده می شود که هر وسیله ممکن است به عنوان یک منبع داده یا سینک داده تصور شود همانطور که در ساختار سیستم پایه در شکل 2 نشان داده شده است. سنسورها و کنترل ها هر دو منبع داده اند. محرکها و ابزارهای مجازی سینک داده ها هستند. تبدیل داده ها و عملیات خودمختار هر دو سینک و منبع هستند ، با سینک مربوط به ورودی به عملکرد و نتیجه پردازش یا خروجی کنترل یک منبع داده است. هر جهت از کانال ارتباطات ممکن است یک سینک داده در انتهای انتقال و یک منبع داده در گیرنده در نظر گرفته شود.

The core of the system is therefore a data manager that coordinates the data received from the sources, and ensures that it gets passed to the corresponding sinks.

بنابراین هسته اصلی سیستم یک مدیر داده است که داده های دریافت شده از منابع را با هم هماهنگ می کند و اطمینان حاصل می کند که آن را به سینک های مربوطه منتقل می کند.



شکل 2: ساختار اساسی سیستم که در آن داده های ورودی منبع داده ای محسوب می شوند و داده های خروجی یک سینک داده است.

* 1. **System Operation**

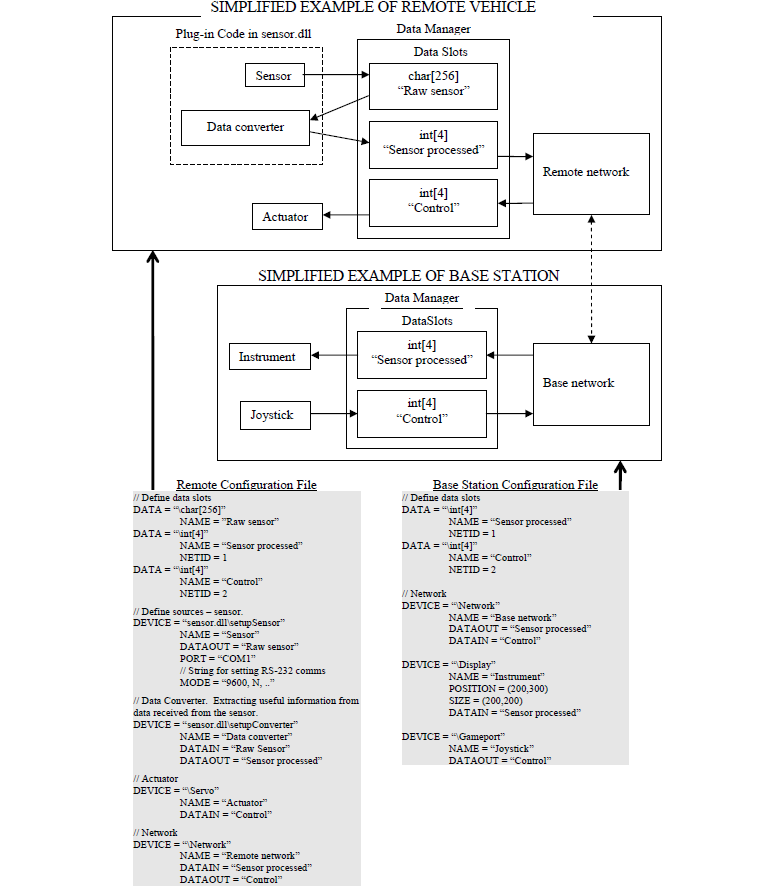
To maintain maximum flexibility, it is important to keep the operations performed on the data separate from the sensing and actuation processes. This may be accomplished by facilitating communication between the various modules through a set of data buffers. These buffers may be thought of as a little like pigeon-holes in a filing clerk’s office.

2.1 عملیات سامانه

برای حفظ حداکثر انعطاف پذیری ، مهم است که عملیات انجام شده بر روی داده ها از فرآیندهای سنجش و فعال سازی جدا شود. این ممکن است با تسهیل ارتباط بین ماژول های مختلف از طریق مجموعه ای از بافر داده ها انجام شود. این بافرها ممکن است کمی شبیه به سوراخ های کبوتر در دفتر منشی تشکیل پرونده باشد.

As new data becomes available at a source (either because a sensor reading is available, or a control input has changed) the new data is placed into a corresponding data slot within the manager. The manager then notifies all of the data sinks that depend on the new data so that they may begin their processing, transmitting, or displaying depending on their particular function. The data processing is facilitated by a task queue. Every data sink must register itself with a data slot that contains its input data. When new data becomes available, the manager adds each of the data sinks that have registered to process the new data into a task queue, where each is called in turn to process the data.

با ورود داده های جدید در یک منبع (یا بخاطر اینکه خواندن سنسور در دسترس است ، یا ورودی کنترل تغییر کرده است) داده های جدید در یک اسلات داده مربوطه در مدیر قرار می گیرند. سپس مدیر به کلیه سینکهای داده وابسته به داده های جدید اطلاع می دهد تا بسته به عملکرد خاص خود ، پردازش ، انتقال یا نمایش خود را آغاز کنند. پردازش داده ها توسط یک صف کار تسهیل می شود. هر سینک داده باید خود را با اسلات داده ای که شامل داده های ورودی آن است ثبت کند. هنگامی که داده های جدید در دسترس می باشد ، مدیر هر یک از سینک داده هایی را که ثبت نام کرده اند برای پردازش داده های جدید در صف کار ، اضافه می کند ، جایی که هرکدام به نوبه خود فراخوانی داده ها را پردازش می کند.



شکل 3: جریان اطلاعات و فایل های پیکربندی در یک مثال ساده. این نشان می دهد که چگونه سنسورها ، مبدل های داده ، ارتباطات دو طرفه ، ابزارها و کنترل ها همه در چارچوب یکپارچه شده اند

Some sensors may require a periodic trigger to acquire data. Within the proposed architecture, this is accomplished by using a timer. The timer also works through the data slot mechanism, in that when a timer times out, this is indicated through a data slot. To be triggered, a sensor only needs to be registered as a sink on the timer data slot. When the timer times out, then the sensor trigger is queued, and when executed, the data capture phase begins.

برای به دست آوردن داده ها ممکن است برخی از سنسورها به یک ماشه دوره ای نیاز داشته باشند. در معماری پیشنهادی ، این با استفاده از یک تایمر انجام می شود. تایمر همچنین از طریق مکانیزم اسلات داده کار می کند ، به این ترتیب که وقتی یک تایمر تمام می شود ، این از طریق یک اسلات داده نشان داده می شود. برای تحریک ، یک سنسور فقط باید به عنوان سینک در اسلات داده تایمر ثبت شود. هنگامی که زمان تایمر تمام میشود ، ماشه حسگر در صف قرار می گیرد و هنگامی که اجرا شد ، مرحله ضبط داده ها شروع می شود.

Keeping data sources and handlers separate in this way enhances flexibility and greatly simplifies reconfiguration. Data dependencies are implicit through the data slots, and this dependency is specified in a configuration file.

جدا نگه داشتن منابع داده و دستگیران از این طریق انعطاف پذیری را تقویت کرده و پیکربندی مجدد را بسیار ساده می کند. وابستگی داده ها از طریق اسلات داده ها ضمنی است و این وابستگی در یک فایل پیکربندی مشخص شده است.

**2.2 System Initialization**

To maintain flexibility, a particular application is set up using a configuration file. This is loaded into the program upon start up. A simple example is shown in Figure 3. There are two sections to the file.

* 1. آماده سازی اولیه سامانه

برای حفظ انعطاف پذیری ، یک برنامه خاص با استفاده از یک فایل پیکربندی تنظیم می شود. این کار پس از شروع به برنامه بارگیری می شود. یک مثال ساده در شکل 3 نشان داده شده است. دو بخش فایل وجود دارد.

The first section defines the data slots. A data slot is identified with the keyword “DATA”. It specifies the type of data contained (char, int), the name, the size and network identification code if it is needed. The name of a data slot must be unique as it is used to identify the data buffer to any devices that may use data from, or produce data for that slot. If the data slot is a timer, there is a field which shows how often that data slot is activated. There is also the facility of creating a custom data slot if needed. Data slots are derived from a common C++ base class that performs the management operations such as signalling when new data is available, and queuing data sinks on the task queue. A custom data slot allows more complex data structures to be implemented. In the example in Figure 3, there are three data slots created in the remote system, and two in the local system. The first is a character buffer that can hold string of up to 256 characters. All of the other slots are arrays of 4 integers.

بخش اول اسلات های داده را تعریف می کند. یک اسلات داده با کلید واژه "DATA" مشخص می شود. در صورت نیاز نوع داده های موجود (char ، int ) نام ، اندازه و کد شناسایی شبکه را مشخص می کند. نام اسلات داده ها باید منحصر به فرد باشد زیرا برای شناسایی بافر داده به دستگاههایی که ممکن است از داده استفاده کنند یا داده برای آن اسلات تولید کند ، استفاده می شود. اگر اسلات داده یک تایمر باشد ، فیلدی وجود دارد که نشان می دهد چند بار آن اسلات داده فعال می شود. در صورت نیاز امکان ایجاد اسلات داده های سفارشی نیز وجود دارد. اسلات داده ها از یک کلاس پایه معمولی C ++ گرفته شده است که در صورت دسترسی به داده های جدید ، عملیات مدیریتی مانند سیگنالینگ را انجام می دهد ، و صف کشیدن داده ها را در صف کار قرار می دهد. یک اسلات داده های سفارشی اجازه می دهد تا ساختار داده های پیچیده تری به مرحله اجرا درآید. در مثال در شکل 3 ، سه اسلات داده ایجاد شده در سیستم از راه دور ، و دو در سیستم محلی است. اولین مورد بافر کاراکتر است که می تواند رشته ای تا حداکثر 256 کاراکتر داشته باشد. همه اسلات های دیگر آرایه های 4 عدد صحیح هستند.

A device is any data source or sink and is defined by the keyword “DEVICE”. The second section of the configuration file specifies the devices. The device keyword specifies the location of the constructor for the C++ class that will handle the device. This may either be a built-in device or contained within a DLL. By making use of DLL plug-ins, whenever a new device is added, the code for handling that device can be easily provided without having to rebuild the complete system.

دستگاه هر منبع داده یا سینک است و با کلید واژه "DEVICE" تعریف شده است. بخش دوم فایل پیکربندی دستگاه ها را مشخص می کند. کلمه کلیدی دستگاه مکان سازنده را برای کلاس C ++ که دستگاه را اداره می کند ، مشخص می کند. این ممکن است یک دستگاه داخلی باشد یا در DLL موجود باشد. با استفاده از افزونه های DLL ، هر زمان که دستگاه جدیدی اضافه شود ، می توانید بدون نیاز به بازسازی سیستم کامل ، کد دستیابی به آن دستگاه را به راحتی تهیه کنید.

The name attribute of the device is not actually required by the system, but provides documentation, and can be used to identify a device in error messages when something goes wrong. The other attributes for each device depend on what type of device it is. Those that are data sources specify the data slot that the new data is to be written to. The configuration file therefore associates a source with the data slot. If a device is a data sink, the configuration file indicates where the data is to come from. This enables the manager to register the device as a listener on the data slot. Many of the remaining attributes are device dependent, and specify additional setup information required by the device handler, such as what port or protocol to use to gather the data from a sensor, or the location of a virtual instrument on the screen.

ویژگی نام دستگاه در واقع مورد نیاز سیستم نیست ، اما مستنداتی را در اختیار شما قرار می دهد و در صورت اشتباه می توانید از دستگاه برای شناسایی دستگاه در پیام های خطا استفاده کنید. ویژگی های دیگر هر دستگاه به نوع دستگاه بستگی دارد. آنهایی که منبع داده اند ، اسلات داده ای را که باید داده های جدید برای آنها نوشته شود ، مشخص می کنند. بنابراین فایل پیکربندی یک منبع را با اسلات داده مرتبط می کند. اگر یک دستگاه سینک داده است ، فایل پیکربندی نشان می دهد که داده ها از کجا آمده است. این مدیر را قادر می سازد دستگاه را به عنوان شنونده در اسلات داده ثبت کند. بسیاری از خصوصیات باقیمانده به دستگاه وابسته هستند و اطلاعات تنظیمات اضافی دیگری را که توسط دستگاه گیرنده دستگاه مورد نیاز است ، مانند آنچه از پورت یا پروتکل استفاده می شود برای جمع آوری داده ها از یک سنسور یا مکان یک ابزار مجازی روی صفحه ، مشخص می کند.

Devices may also specify a task queue to use instead of the default. This will be described in more detail later.

دستگاه ها همچنین ممکن است یک صف کار را برای استفاده به جای پیش فرض مشخص کنند. این بعداً با جزئیات بیشتر توضیح داده خواهد شد.

A data converter is just another type of device, which processes the data to extract useful or required information into a more convenient form. For example to convert a string to an integer, or extracting latitude and longitude information from a NMEA string produced by a GPS device [5]. The DATAIN attribute points to the data slot containing the raw information from the sensor, and the DATAOUT attribute points to the data slots being written to with the processed data. Data converters may also be used for providing data compression (for example of speech or video signals). Keeping the converter separate from the source enables different algorithms to be slotted in and out with relative ease.

مبدل داده ها فقط نوع دیگری از دستگاه است که داده ها را برای استخراج اطلاعات مفید یا موردنیاز به فرم راحت تر پردازش می کند. به عنوان مثال برای تبدیل یک رشته به عدد صحیح یا استخراج اطلاعات طول و عرض جغرافیایی از یک رشته NMEA تولید شده توسط دستگاه GPS. ویژگی DATAIN به اسلات داده حاوی اطلاعات خام سنسور اشاره می کند ، و ویژگی DATAOUT به اسلات های داده ای که با داده های پردازش شده نوشته می شوند ، اشاره می کند. مبدل های داده نیز ممکن است برای تهیه فشرده سازی داده ها (به عنوان مثال سیگنال های گفتاری یا فیلم) استفاده شوند. جدا نگه داشتن مبدل از مبدأ ، الگوریتم های مختلفی را با سهولت نسبی در داخل و خارج میسر می سازد.

Autonomous control operations may also be implemented as data converters. In some applications, critical control loops may become unstable with the communication delays between the remote and base station. In such situations, one may decide to implement a local control loop on the remote platform. The sensor data will be used to directly control the actuators, thereby creating an autonomous remote station.

عملیات کنترل خودمختار نیز ممکن است به عنوان مبدل داده ها اجرا شود. در برخی از برنامه ها ، ممکن است با تأخیر در برقراری ارتباط بین ایستگاه از راه دور و پایه ، حلقه های کنترل بحرانی ناپایدار شوند. در چنین شرایطی ، فرد ممکن است تصمیم به اجرای یک حلقه کنترل محلی بر روی سکوی راه دور بگذارد. از داده های سنسور برای کنترل مستقیم محرک ها استفاده می شود و از این طریق یک ایستگاه از راه دور مستقل ایجاد می شود.

**2.3 Data Communication**

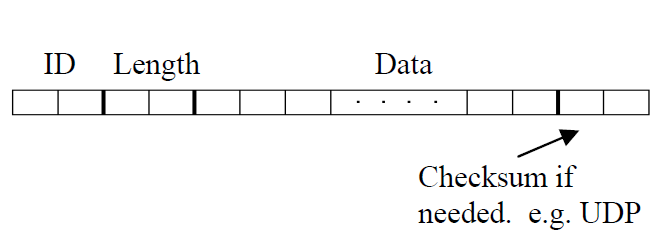
Data communication between the remote and base station may be achieved by any available form of communication. In the prototype, we are using a TCP/IP network.

* 1. ارتباط داده

ارتباط داده بین ایستگاه راه دور و پایه ممکن است با هر شکل ارتباطی موجود حاصل شود. در نمونه اولیه ، ما از یک شبکه TCP / IP استفاده می کنیم.

Network communication starts with the communication module registering itself as a sink on all of the data slots that contain data to be communicated between stations. Each communication data slot is supplied with a network ID used to identify the data to the receiver. When new data is available and the transmit task is triggered, it sends a data packet to the receiver. When data is received, the packet identification is used to determine which data slot the data is placed in at the destination. This activates dependent data sinks at the base station (such as virtual instruments). The same mechanism is used for control upstream, from the base station to the remote vehicle.

ارتباطات شبکه از طریق ماژول ارتباطی که خود را به عنوان سینک در تمام اسلاتهای داده که حاوی داده هایی است که باید بین ایستگاه ها ارتباط برقرار کنند ، شروع می شود. هر اسلات داده ارتباطی با یک شناسه شبکه که برای شناسایی داده ها به گیرنده استفاده می شود ، تهیه می شود. هنگامی که داده های جدید موجود است و کار انتقال انجام می شود ، یک بسته داده را به گیرنده می فرستد. هنگامی که داده ها دریافت می شوند ، از شناسایی بسته استفاده می شود تا داده ها را در قسمت مقصد قرار دهند. این کار سینک داده های وابسته را در ایستگاه پایه (مانند ابزارهای مجازی) فعال می کند. از همین مکانیسم برای کنترل بالادست ، از ایستگاه پایه تا وسیله نقلیه از راه دور استفاده می شود.



شکل 4: ساختار بسته برای برقراری ارتباط بین ایستگاه پایه و ایستگاههای از راه دور. هر جعبه یک بایت را نشان می دهد.

In the data driven approach, the data from each slot is communicated as a network packet, as shown in Figure 4. The structure of the packets has been defined so that the network ID is at the start of the packet. This is two bytes in length, allowing up to 65,536 separate data slots to be linked between the remote and base stations. The length field is next having two bytes. This indicates the length of the data field. Lastly there is the data field which can be up to 65,535 bytes in length. The data format is application defined. For this reason it is important that both remote and base stations have the same representation for integers (big endian / little endian), floating point, etc. Some communication channels, such as TCP, include error checking and correction, so will not need any checksums. However, some communication protocols, such as UDP, will need this. In such cases, a CRC-16 checksum will be added to the end of the packet.

در رویکرد محور داده ، داده های هر اسلات به عنوان یک بسته شبکه ارتباط برقرار می کنند ، همانطور که در شکل 4 نشان داده شده است. ساختار بسته ها به گونه ای تعریف شده است که شناسه شبکه در ابتدای بسته است. طول این دو بایت است که به 65536 اسلات داده جداگانه اجازه می دهد تا بین ایستگاه راه دور و پایه متصل شوند. قسمت طول دارای دو بایت است. این طول فیلد داده را نشان می دهد. در آخر ، قسمت داده وجود دارد که می تواند طول آن تا 65535 بایت باشد. قالب داده ها برنامه تعریف شده است. به همین دلیل مهم است که هر دو ایستگاه از راه دور و پایه دارای نمایشگر یکسان برای اعداد صحیح (بزرگ endian / کمی کوچک endian) ، نقطه شناور و غیره باشند. بعضی از کانالهای ارتباطی مانند TCP شامل بررسی خطا و تصحیح هستند ، بنابراین نیازی به هیچ چک با این حال ، برخی از پروتکل های ارتباطی ، مانند UDP ، به این امر احتیاج دارند. در چنین مواردی ، یک چک CRC-16 به انتهای بسته اضافه می شود.

Upon initial connection of the network, network control packets will pass between the remote and base station. The purpose of this is to check that the data slots at each end of the link are of the same type and have the same name and network identification. Any errors on either end are reported to the base station for the user.

پس از اتصال اولیه شبکه ، بسته های کنترل شبکه بین ایستگاه از راه دور و پایه عبور می کنند. هدف از این کار بررسی این است که اسلات های داده در هر انتهای لینک از یک نوع برخوردار بوده و دارای نام و شناسایی شبکه یکسان هستند. هر گونه خطا در هر دو انتها برای کاربر به ایستگاه پایه گزارش می شود.

**3 Implementation Issues**

**3.1 Class Structure and use of DLLs**

**3.2 Multithreading and Task Queues**

**3.3 Communications**

**4 Summary**

<https://pdfs.semanticscholar.org/8613/4435062a6c00509077c4953d328b631719ee.pdf>