Arduino for the Cloud

Arduino for the Cloud

Arduino Yún and Dragino Yún Shield

Claus Kühnel

Universal-Publishers
Boca Raton

Arduino for the Cloud: Arduino Yún and Dragino Yún Shield

Universal-Publishers
Boca Raton, Florida • USA
2015

ISBN-10: 1-62734-035-1
ISBN-13: 978-1-62734-035-9

www.universal-publishers.com

The Linux Penguin is a work of authors
Larry Ewing, Simon Budig and Anja Gerwinski.

Preface

Raspberry Pi and comparable computer modules use embedded Linux as an operating system with the result that embedded Linux is no longer reserved for specialists alone, but can be used by amateurs and enthusiasts as well. At the same time, a way has been opened to start with a class of completely new tasks. The base for communication among several components has been laid and we are able to develop the Internet of Things (IoT).

Wikipedia provides a comprehensive description of the IoT. Embedded systems with limited resources can collect data from natural ecosystems as well as from buildings and factories. Otherwise, these systems could also be responsible for performing actions like ordering missing goods in a storage system. Energy management and transport optimization, as heard of these days at the Port of Hamburg, are further examples.

In the paper entitled "The Computer for the 21st Century" [1], Mark Weiser (http://en.wikipedia.org/wiki/Mark_Weiser) spoke about this vision for the first time in 1991.

Today, the Arduino family comprises more than 20 boards (not including Arduino clones) and creates a largely mature platform for building professional prototypes. The unusual concept of open-source hardware and software was awarded the Prix Ars Electronica in Linz in 2006.

If we look into the relevant forums and portals covering microcontroller technology, we cannot miss the topic of Arduino.

Google provided a further impulse through its decision to use Arduino as the Android Open Accessory kit [2]. Arduino is an open-source prototyping platform based on flexible and easy-to-use hardware and software.

The primary goal for using Arduino is direct contact with the environment. Arduino can detect signals from sensors and control this environment via several actors. The term "physical computing" describes the kind of applications for which access to the environment is an important attribute.

The high complexity of today's requirements for electronic systems cannot be implemented by concepts based on classic microcontrollers. In most cases, the networking of components is indispensa-

ble. In the absence of these preconditions, the IoT would be wishful thinking.

Arduino Yún combines the classic Arduino based on Atmel's AVR microcontrollers with an Atheros AR9331 System-on-a-Chip (SoC). This SOC is used in WLAN access points and routers, and runs the Linux distribution Linino (OpenWRT) as operating system.

The family of Arduino shields has recently been extended by Dragino's Yún shield. The combination of the Dragino Yún shield with an Arduino Leonardo can be compared with an Arduino Yún.

While an Arduino Leonardo with the Dragino Yún shield together is functionally identical to Arduino Yún, the Yún shield offers more flexibility through possible combinations with Arduino Uno, Duemilanove, Mega, etc. In addition, the Dragino Yun shield uses an external WiFi antenna that promises more stability and robustness of the wireless connection.

The operating system Linino (OpenWRT) offers interface drivers, a file system and multi-threading, among other things, and is responsible for recurring tasks using stable software components.

This book is a translation of "Arduino für die Cloud" (ISBN 978-3-907857-25-0), which was published in German. It deals with the Arduino Yun and the combination of Arduino Leonardo with the Dragino Yun shield and considers both implemented controllers and their interaction.

All listed sources and some explanations can be found at SourceForge under http://sourceforge.net/projects/arduinoynsnippets/.

For ease of readability, I follow in the textual representation the conventions listed below:

- Commands and output to the console are represented in `Courier New`.
- Inputs via console are in **`Courier New`**.
- Program and file names appear in *Italian*.

All existing links were checked in the autumn of 2014. As the internet changes continuously, it cannot be assured that these links will work or lead to the same content as at the time of admission. Please inform me about broken links.

Altendorf, spring 2015
Claus Kühnel

Content

1 Introduction

If you want to extend an Arduino so that it is network enabled, then you could use, for example, an Arduino Uno supplemented with an Ethernet shield or you could use an Arduino Ethernet. The Ethernet shield and the Arduino Ethernet provide an Ethernet interface based on the WIZnet Hardwired TCP/IP Embedded Ethernet Controller W5100.

Figure 1 shows an Ethernet shield that can be placed on an Arduino and provides them an Ethernet interface.

Figure 2 shows an Arduino Ethernet, a combination of an Arduino and an Ethernet interface on one board. The difference from other Arduino boards is the missing USB-to-serial chip. The Ethernet part is identical to the Ethernet shield.

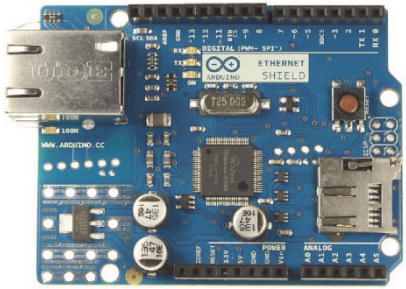

Figure 1 Ethernet shield

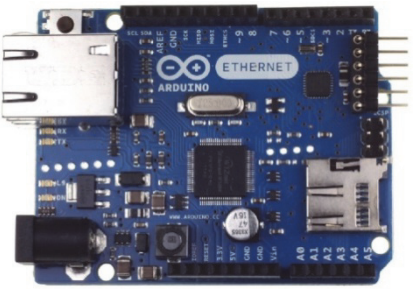

Figure 2 Arduino Ethernet

The Arduino Yún differs from other Arduino boards through its capabilities to communicate with the Linux system running on the Atheros AR9331. These capabilities make the Arduino Yún a powerful platform for Linux application in a network and IoT projects combined with the simplicity of the Arduino. Additionally to powerful Linux commands like *cURL*, an option is to use one's own Shell or Python scripts for a robust interaction with Arduino Yún.

The Arduino Yún is similar to the Arduino Leonardo because Arduino Leonardo uses an ATmega32u4 too. Since the ATmega32u4 has an integrated USB controller, it has no need for a second controller, like an FT232 for example. The Arduino Yún represents itself

against a connected computer additionally to the virtual (communication device class) COM port as mouse, keyboard or another HID interface.

Figure 3 shows the front side of an Arduino Yún. In the upper part, you can see the Atheros AR9331, a 2.4 GHz System-on-a-Chip (SoC) developed for WLAN and router applications.

Figure 4 shows the rear of an Arduino Yún with an AU6350 Single-Chip-USB 2.0 hub and multimedia card reader controller and an SD card holder.

On the rear, you can see the internationality of this product. The fathers of the Arduino are from Italy. Dog Hunter, which is located in the U.S.A., developed the board. The layout work was done in Switzerland and production in Taiwan.

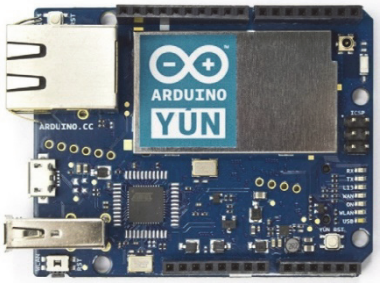

Figure 3
Arduino Yún—front view

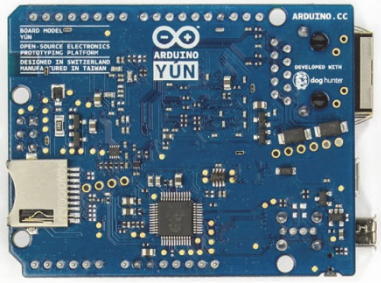

Figure 4
Arduino Yún—rear view

Launched by the Chinese company Dragino (http://www.dragino.com), the Dragino Yún shield includes all the components of the Arduino Yun without the ATmega32u4. The Dragino Yún shield gets the Arduino Yún functionality by connecting it to an Arduino Leonardo. But it is possible to connect alternative Arduino boards with the Dragino Yún shield, too.

The encapsulated Dragino HE Module is the core of the Dragino Yún shield. Its Atheros AR9331 SoC runs the same Linino (OpenWRT) as Linux distribution, as for Arduino Yún. Figure 5 shows the top view of the Dragino Yun shield.

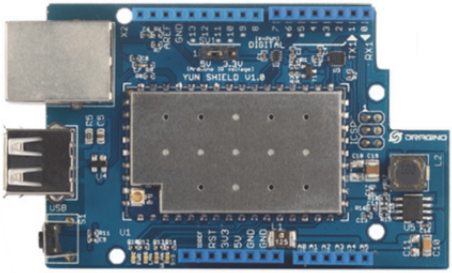

Figure 5 Dragino Yún shield

This book explains the start-up of the Arduino Yún and the Dragino Yún shield connected to an Arduino Leonardo and their use. These explanations should provide a quick overview of the enhanced possibilities of these LAN- and WLAN-capable Arduinos.

Detail information about the Arduino Yún and the Dragino Yún shield can be found via the following URLs:

- http://arduino.cc/en/Main/ArduinoBoardYun
- http://www.dragino.com/products/yunshield/item/86-yun-shield.html

2 Arduino Yún

This chapter describes the similarities and differences between the Arduino Yún and the Dragino Yún shield. Hence, possible hurdles are largely eliminated when using one of the two platforms.

2.1 Power Supply

2.1.1 Arduino Yún

The micro-USB interface is a common way to connect an Arduino with a power supply. Furthermore, this connection serves as a communication path to the Arduino development environment (IDE).

The Arduino Yún has no voltage regulator onboard; therefore, the voltage at pin VIN has to be 5V DC constant. Pay attention to this requirement when choosing the power supply.

Another possibility is to source the Arduino Yún via the Ethernet. The board is prepared for the use of a Power over Ethernet (PoE) module.

Power supply details:

VIN	External connection for supply voltage of 5 V DC
5V	Internal 5 V supply voltage Voltage source can be VIN or USB
3V3	Internal 3.3 V supply voltage, generated by an on-board regulator. Load can be 50 mA
GND	Ground
IOREF	Voltage of the I/O pins (VCC = 5 V)

2.1.2 Dragino Yún Shield

The Dragino Yún shield gets its supply voltage from the connected Arduino board.

The Dragino HE draws under full load approximately 200 mA. Its supply voltage is derived from the voltage at VIN of the Arduino board and will be converted on the Dragino Yún shield itself to 3.3 V DC. The Dragino Yún shield should be powered under these conditions via a VIN voltage of the Arduino board between 7 and 15 V DC and not via the USB connection.

The supply voltage for the USB host is derived from +5 V DC of the Arduino. To avoid overheating of the voltage regulator on Arduino when using the USB host, the voltage at VIN should be limited to 7 V DC.

When using the Dragino Yun shield, I have always worked with a voltage of 7 V DC on pin VIN. The average current consumption was 0.15 A in this case.

2.2 Memory

2.2.1 Arduino Yún

The ATmega32u4 used on Arduino Yún has a 32 KB flash memory. The boot loader uses 4 KB of this memory. The ATmega32u4 provides 2.5 KB of RAM and 1 KB of EEPROM for use. The EEPROM library supports reading and writing of that EEPROM.

The Atheros AR9331 has no internal memory. On board is 64 MB of DDR2RAM and a 16 MB flash memory. The Linux distribution Linino (OpenWRT) is factory-installed in the flash memory. This factory image can be changed. A reset to the factory-installed image can be achieved by pressing the key WLAN RST for 30 seconds.

The existing memory can be enhanced by a micro-SD card or a USB memory stick. SD card and USB connectors are available on board.

2.2.2 Dragino Yún Shield

The Dragino Yún shield also has a 64 MB RAM and a 16 MB flash memory but the memory on the Arduino side is defined by the connected Arduino board.

I use here an Arduino Leonardo and the ATmega32u4 used on Arduino Leonardo provides 32 KB of flash memory once again (4 KB are occupied by the boot loader) as well as 2.5 KB RAM and 1 KB EEPROM. The EEPROM library supports reading and writing of the EEPROM again.

2.3 I/O

2.3.1 Arduino Yún I/O

All I/O pins on Arduino Yún that can be connected externally are pins of the ATmega32u4 microcontroller. The programming of these pins occurs Arduino-conform with the instructions `pinMode()`, `digitalWrite()` and `digitalRead()`.

All pins operate with 5 V (defined by IOREF) and can source or draw a current of 40 mA pro pin maximum. Pull-up resistors are available in a range between 20 and 50 k**Ω**, but these are disabled by default.

Several pins have alternative functions:

Function	*Pin*	*Description*
Serial	0 (RX) 1 (TX)	Receiving (RX) and transmitting (TX) serial data with TTL level from/to the ATmega32u4 hardware interface. The class **Serial** refers to the USB (CDC) interface. The class **Serial1** refers to the TTL communication to the AR9331 via pins 0 and 1.
TWI(I^2C)	2 (SDA) 3 (SCL)	I^2C interface The Wire library supports this communication.
External Interrupts	0 (INT2) 1 (INT3) 2 (INT1) 3 (INT0) 7 (INT4)	These pins can be used as interrupt lines. The pins can be configured to the following interrupt requests: Low at pin, rising or falling edge or change of the level. Details are described at function `attachInterrupt()`. Pins 0 and 1 should not be used as inter-

		rupt line, because these lines are used for serial communication between ATmega32u4 and AR9331. Pin 7 is connected to the AR9331 too and could be used as handshake line later. Therefore, only pins 2 and 3 are available for unscrupulous use as interrupt lines, if you do not use the I²C interface.
PWM	3, 5, 6, 9, 10, 11, 13	8-bit PWM output using the function `analogWrite()`.
SPI	ICSP Header	SPI data transfer using the SPI library. The SPI pins of Arduino Yún are connected to the ICSP header only. This can mean restrictions for the use of Arduino shields. The SPI pins are connected to the AR9331; hence, ATmega32u4 and AR9331 can transfer data via SPI too.
LED	13	This is the well-known internal LED connected to pin 13 onboard (Hi = LED on, Lo = LED off).
Analog Input	A0–A5, 4 (A6) 6 (A7) 8 (A8) 9 (A9) 10(A10) 12(A11)	Arduino Yún has 12 analog inputs (A0–A11) with a resolution of 10 bits. Per default, the input voltage range is 0–5 V. This range can be reduced by the use of the AREF pin and the function `analogReference()`.
AREF		Reference voltage for analog inputs. Usage with the function `analogReference()`.
Yún RST		A low level on this pin resets the AR9331. The Linux system will be rebooted. All data in the RAM are lost. Running programs will be stopped. Files can be corrupted.
32U4 RST		A low level on this pin resets the ATmega32u4.

WLANRST		The primary function is to reset WiFi to the factory setup. The factory setup is the Access Point Mode (AP) and the fix IP address 192.168.240.1. In this mode, the Arduino Yún can be configured. A reset of the WiFi configuration forces a reboot of the Linux system. To reset the WiFi configuration, press the key WLAN RST for five seconds. The secondary function is to reset the Linux system to the factory image. To reset the Linux system, press the key WLAN RST for 30 seconds. All data stored in the flash memory is lost.

Additionally, the Arduino Yún has some LEDs onboard:

Marking	*Function*
RX	Serial receiver
TX	Serial transmitter
L13	LED at IO13
WAN	WAN (Ethernet) indicator
ON	Power indicator
WLAN	WLAN (WiFi) indicator
USB	USB

2.3.2 Dragino Yún Shield I/O

The Dragino Yún shield does not make any I/O pins available outwards. In the combination of Arduino Board and the Dragino Yun shield, the available I/O pins come solely from the Arduino board.

Differences from the Arduino Yún:

S1:	S1 serves as reset key. Pressing it for five seconds resets the WiFi configuration. All other settings remain unchanged. Pressing it for 30 seconds resets the Linux system to the factory image.
SV1:	The jumper SV1 configures the voltage level for SPI and UART (5 V and 3.3 V respectively). Arduino Leonardo works with 5 V levels.

Additionally, the Dragino Yún shield has some LEDs onboard:

Marking	*Function*
PWR	Power indicator
LAN	LAN (Ethernet) indicator
WLAN	WLAN (WiFi) indicator
SYS	LED for USB memory. On = USB is connected to the Arduino Yun default SD directory /mnt/sd or /www/sd connected.

2.4 Communication

2.4.1 Arduino Yún

The Arduino Yún has several interfaces for communicating with a computer, another Arduino or another microcontroller.

The ATmega32u4 provides UART hardware with TTL levels for serial communication. This interface (IO0 and IO1) is used for serial communication between the ATmega32u4 and the Atheros AR9331. The Bridge library supports the software side.

In addition, the ATmega32u4 offers the possibility of serial communication over USB and is then viewed from a connected PC as a virtual COM port. The software download from the PC to the Arduino usually occurs through this interface equipped with a micro-USB connector. This interface can also be used for monitoring. The data transfer through this interface is indicated by the flashing of the RX and TX LEDs on the board.

Using the Software Serial Library, software UART's can be implemented, so that each pin of the ATmega32u4 can be used for serial

communication. But IO0 and IO1 are reserved for the hardware UART. The ATmega32u4 supports I²C and SPI communication as well. The Wire library supports communication via the I²C bus and the SPI library that for the SPI bus.

Arduino Yún can be programmed as a generic keyboard and mouse. For this, keyboard and mouse classes are available.

The Atheros AR9331 provides the Ethernet and WiFi interfaces. The Bridge library is responsible for this communication too.

The Arduino Yún offers through Linux support, another USB host interface. Via this USB port, we can connect a memory stick for memory enhancement, a keyboard, a mouse, and a webcam. It may be necessary to install other software packages. I will return to this later.

2.4.2 Dragino Yún Shield

The Dragino HE module uses SPI and UART to communicate with the connected Arduino board. The Dragino Yún shield is compatible with 3.3 V and 5 V Arduino boards. The jumper SV1 on the Dragino Yún shield selects the used voltage level (3.3 V or 5 V).

The program upload from Arduino IDE uses the SPI interface and connects both controllers. SPI is free for other SPI slaves in an application program after this upload.

The UART interface (IO0 and IO1) serves as a bridge between both controllers as in Arduino Yún.

2.5 Initial Startup

The statements for the initial startup refer to the Arduino Yún and the combination of Arduino Leonardo—Dragino Yún Shield alike. I use here an Arduino Leonardo for the combination with the Dragino Yún shield, because it can be connected without additional modifications.

The steps required for connecting an Arduino Uno, Duemilanove/Diecimila as well as Mega2560 are described in detail in the Yun Shield User Manual—version 1.1; therefore, the link to the user manual,
http://www.dragino.com/downloads/downloads/YunShield/YUN_SHIELD_USER_MANUAL_v1.0.pdf, seems sufficient here. In addition, the relevant boards are to be considered in the software configuration.

Before the initial startup of the Arduino Yún, the Arduino Development Environment must be downloaded from the Arduino website to the computer used for software development.

I use here a development PC running Microsoft Windows, for which you will find the installer at the URL http://downloads.arduino.cc/arduino-1.5.7-windows.exe. Installers for Linux and MacOS can be found in the same place.

Arduino 1.6.1 with several fixes and improvement for Arduino Yún also was released on 03/10/2015. Take this revision for new developments, although the program samples published here were tested using Arduino 1.5.7.

The Arduino Yún is only supported as of version 1.5.4 of the development environment, which is important to remember before download.

Arduino enthusiasts, not unexpectedly, will get the known user interface to face. However, there is some news here. Figure 6 shows the list of supported Arduino boards, including our Arduino Yún at the top of the list.

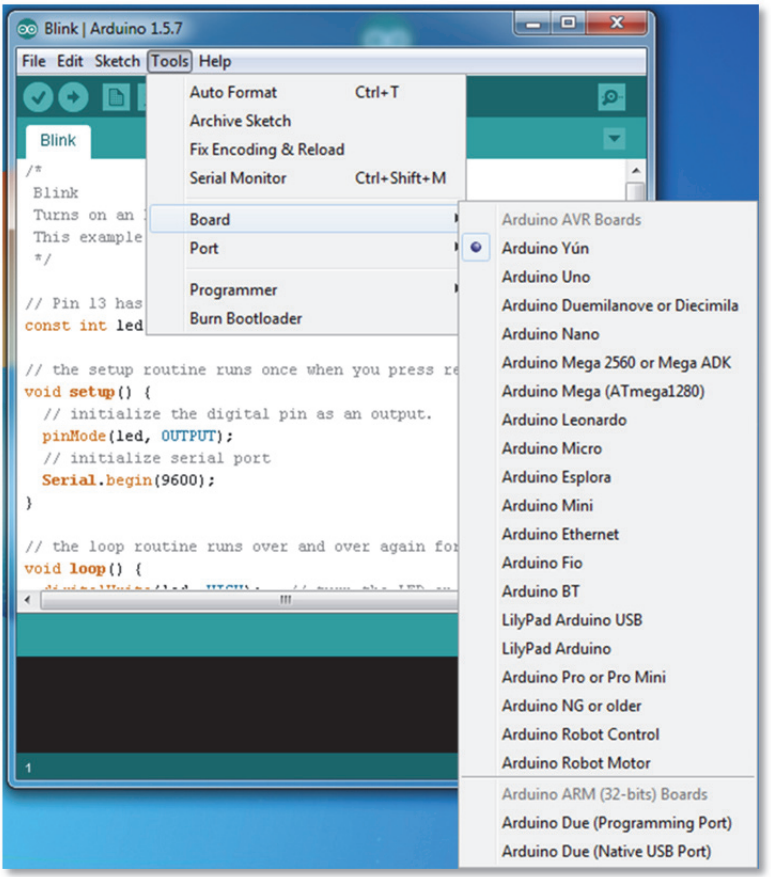

Figure 6 Arduino boards in the Tools menu

After connecting the micro-USB connector of the Arduino Yún with an USB port of your PC, the drivers install automatically and you can find the used COM port with the help of the device manager.

Figure 7 shows that our Arduino Yún is connected to the (virtual) port COM16 and it is possible to communicate via USB with the PC.

There is another interface named "myYUN at 192.168.1.29 (Arduino Yún)", which I will explain later.

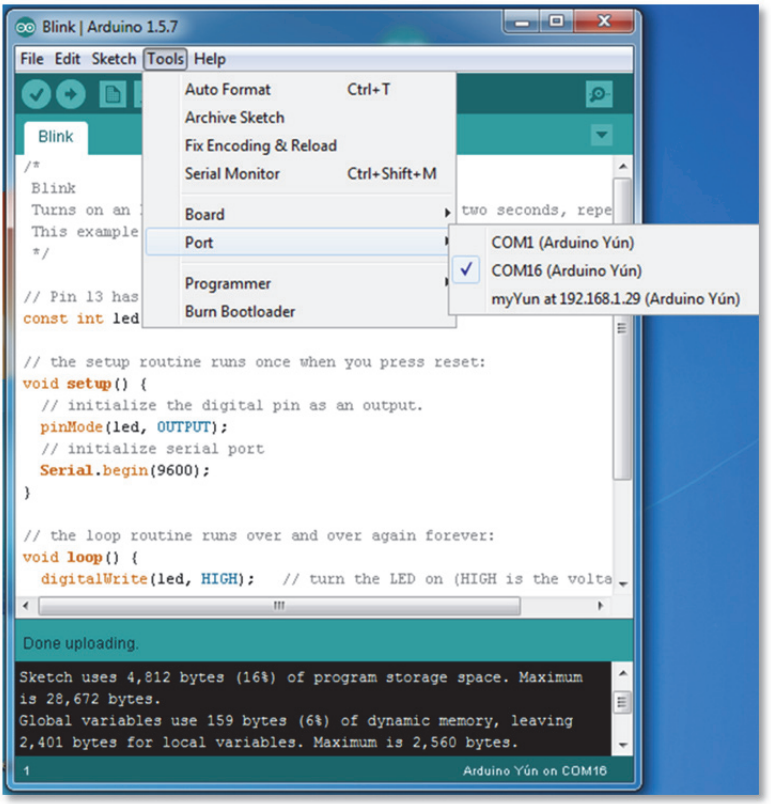

Figure 7 Arduino Yún communication interfaces

The Arduino program *blink.ino*, which may be inspected in the editor, can be uploaded via COM16 into the Arduino Yún.

This program provides a blinking red LED and periodic output of the character * via the USB port, which can be displayed in a monitor program (Figure 8). The baud rate of the monitor program must be equal to the baud rate defined in the function `setup()`.

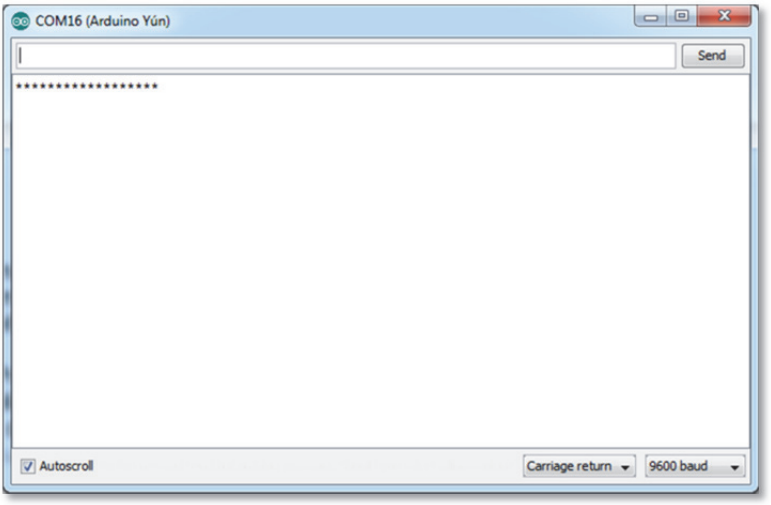

Figure 8 Monitor output

For those who already have experience in programming Arduino, these steps are not new. For the person who is using Arduino for the first time, the Arduino website http://arduino.cc offers good support.

There is also a lot of literature explaining Arduino now. Amazon.com shows 1,128 results for books with "Arduino" in the title. Please use your preferred search engine to find your solution.

As of Version 1.0 of the Arduino IDE, some changes have been introduced, which must be considered when porting programs from previous versions.

2.6 Architecture

On the Arduino Yun, two worlds meet. Figure 9 shows a block diagram of the Arduino Yún (http://arduino.cc/en/Main/ArduinoBoardYun).

On the left side, we see the Arduino based on an ATmega32u4 (http://www.atmel.com/devices/ATmega32u4.aspx) with a micro-USB connector for programming and communication with the Arduino IDE, which can be used to power the Arduino Yun simultaneously.

On the right side, we see the Linux device based on an Atheros AR9331
(http://www.datasheetspdf.com/PDF/AR9331/743003/24). The

Linux device is running Linino (http://linino2013.wordpress.com/about/) derived from OpenWRT (https://openwrt.org/).

With an Ethernet interface (eth1) and a wireless interface (wlan0), the Atheros AR9331 provides opportunities for integration into a network. In addition, an SD card interface and a USB host port are available for external expansion.

The Bridge library makes the communication between Atheros AR9331 and ATmega32u4 easier. We get the opportunity to run shell scripts, communicate with the network and obtain information from the AR9331 processor for an Arduino sketch out. The USB host as well as the network interface and the SD card are not connected to the ATmega32u4 but with the AR9331. The Bridge library also allows the ATmega32u4 to access the AR9331 periphery.

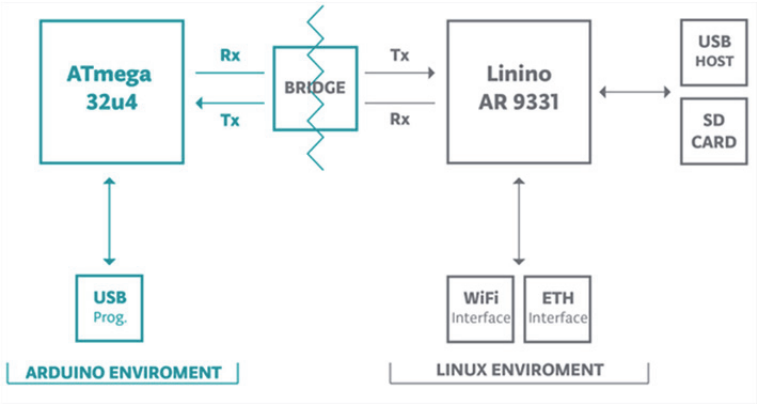

Figure 9 Arduino Yún block diagram

At Arduino Yún, the focus remains on the Arduino. However, Arduino enthusiasts get the opportunity to use web services over wired or wireless Ethernet, without deep knowledge of Linux. Due to the Atheros AR9331, which takes over the entire network management, the Linux side is encapsulated, but remains fully available. Some technical features of the Arduino Yún are summarized in Table 1. The part for the Atheros AR9331 applies to the Dragino Yún shield too, while the part for the ATmega32u4 applies for the Arduino Yún only. For the combination of Dragino Yún shield with an Arduino, the technical data of the used Arduino apply.