An open-source multi-robot construction system

Michael Allwright *, Weixu Zhu, Marco Dorigo

Université libre de Bruxelles, Belgium

ARTICLE INFO

ABSTRACT

We describe a completely open source system for performing experiments in multi-robot construction in laboratory settings. The system consists of robots that are capable of assembling cubic blocks into structures, which can be up to three blocks in height. The building material contains microcontrollers and multi-color light-emitting diodes (LEDs) that can be programmed by the robots using a near-field communication (NFC) interface. This mechanism is implemented to facilitate experiments where the intelligence that coordinates the construction can be embedded not only in the robots but also in the building material.

© 2018 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Specifications table:

Hardware name	SRoCS (Swarm Robotics Construction System)
Subject area	Engineering and Material Science
Hardware type	Electrical engineering and computer science
Open-source licenses	Hardware and high-level software: MIT License
	Microcontroller firmware: GNU LGPL 2.1
	Linux kernel modules: GNU GPL 2
Cost of hardware	36,500.00 EUR (5 Robots, 50 Blocks)
Project repository	https://osf.io/spfx7/

1. Hardware in context

We present the hardware of our open-source robotics construction system for experiments in multi-robot construction, which we call the Swarm Robotics Construction System (SRoCS). We have designed SRoCS as a research tool for studying how swarms of robots [8] can be programmed to assemble structures collectively. The system is designed to operate in laboratory settings and consists of two components, a building material and a mobile robot capable of assembling that building material into structures.

Using the hardware described in this article, we intend to study how the behavior and coordination mechanisms used by social insects during nest construction can be implemented in a multi-robot construction system. An example of such behavior is exhibited by social wasps, which coordinate the construction of their nest by changing the shape of combs as they are built [12]. A further example is exhibited by termites, which coordinate the construction of their nest's royal chamber by

---

* Corresponding author.
  E-mail address: michael.allwright@ulb.ac.be (M. Allwright).

using pheromones to mark soil pellets [6]. The construction in these two examples is coordinated through *stigmergy*, a form of indirect communication where an individual insect modifies its environment via its work and where those modifications then regulate the work of other insects [18]. Stigmergy was first observed in nature by Grassé during his work with termites [9]. The application of the coordination mechanisms used by social insects to multi-robot construction is attractive since construction by social insects has been shown to be robust, parallel, and adaptive to the environment in which it is situated [5,7].

To simulate the coordination mechanisms used by social insects in our system, we have implemented an advanced building material that we call the Stigmergic Blocks. The Stigmergic Blocks contain multi-color light-emitting diodes (LEDs) that can be configured using near-field communication (NFC) by the robots. We have shown in previous work that this functionality of our hardware makes it possible to assemble a target structure by encoding the building process in terms of the structural arrangement and LED colors at the intermediate stages of the target structure's construction [1].

The presented system has been designed to study how structures can be assembled by swarms of robots. As such, our experimental focus is on developing reactive, decentralized control algorithms for construction. Our robots are designed to perform the necessary computer vision on board. While it is possible to enhance the capabilities of the system by giving the robots access to global sensory and computational resources (for example, an overhead camera system or a remote server for offloading image processing), such configurations are beyond the scope of our experimental focus and have not been tested on the presented hardware. For examples of multi-robot construction systems that use global sensory or computational resources, we refer the reader to the work of Augugliaro et al. [4], Lindsey et al. [13], and Worcester et al. [20]. To the best of our knowledge, there are three other systems that use a decentralized control strategy to achieve multi-robot construction: (i) the work by Petersen et al. and Werfel et al. [15,19], (ii) the work by Jones and Matarić [11], and (iii) the work by Sugawara and Doi [17]. Of these systems, only the work by Petersen et al. and Werfel et al., that is, the TERMES system, achieves multi-robot construction in three dimensions. The approach used in the TERMES system, however, is significantly different from our approach in that the TERMES approach uses odometry to enable a simple robot to track its location on a blueprint of the structure being built. In contrast to this approach, we attempt to more closely replicate the flow of information that occurs during social insect construction and rely exclusively on locally available information primarily gathered using computer vision.

This article provides a description of SRoCS and the instructions to assemble, program, and perform experiments using the hardware. The hardware components of our system are available under an MIT license and are hosted as part of a project on the Open Science Framework. The microcontroller firmware for our system is available under an LGPL license since it is a derivative of the Arduino Core libraries. The drivers for the Linux operating system on the robot are available under a GPL license since they must access parts of the Linux kernel API that are only available to GPL-licensed kernel modules.

2. Hardware description

The SRoCS system consists of two components, a mobile robot called BuilderBot and a building material called the Stigmergic Blocks. BuilderBot evolved from an extension to the BeBot mobile robotics platform [10]. In contrast to BeBot, however, BuilderBot is completely open-source and can be assembled using 3D printed parts and off-the-shelf components. The BuilderBot is capable of building structures that are up to three blocks in height. Due to the reach of the robot, blocks can only be placed directly in front of the robot, which in turn requires structures to be built from the inside out. At the time of writing, the robot can only add blocks to a structure and can not remove them. To the best of our knowledge, the presented hardware is the first completely open-source platform for performing research in autonomous construction. In the following sections, we describe the electronics, mechanical design, and software of BuilderBot and of the Stigmergic Block.

2.1. Stigmergic Block

A Stigmergic Block is an advanced cubic building material capable of computation, data storage, and communication (Fig. 1). The block supports three types of communication: (i) robot-to-block and block-to-block communication using a near field communication (NFC) interface, (ii) block-to-robot communication from a block's LEDs to a robot's camera and (iii) block-to-robot, robot-to-block, and block-to-block communication using an optional Xbee module. An AprilTag is attached to each face of a Stigmergic Block, enabling a robot to reliably estimate its pose relative to a block using the AprilTag detection algorithm [14]. Although the tags on the blocks can have unique identifiers, in our current experiments we use the LEDs on the blocks to differentiate the blocks based on their color. This has the advantage that the colors, which are programmable via the NFC interface, can be changed by the robots during the experiments. In the future, however, we envision experiments where different tags could be used to distinguish blocks with differing functionality (e.g., communication using an Xbee radio) or with differing physical characteristics (e.g., weight).

A freely-rotating, spherical magnet is located in each corner of a block to enable self-alignment and to reduce cumulative misalignment during experiments. These spherical magnets also increase the structural integrity of the structure being built.

In addition to being used with the BuilderBot as a building material, the Stigmergic Blocks could be a useful tool for investigating and running experiments with smart structures. Such structures could be made up of a number of Stigmergic Blocks and could route information between them using their NFC interfaces. For example, the blocks could be used to visualize and
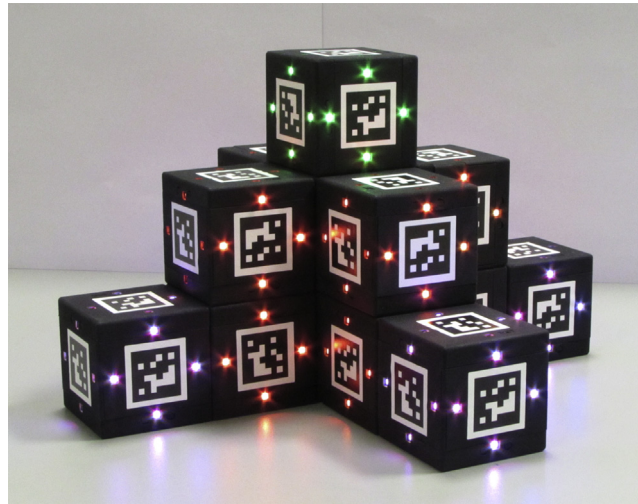
Fig. 1. A structure built from Stigmergic Blocks.

validate routing algorithms in an interactive, physical system, where the nodes in a network can be added and removed by hand.

2.1.1. Electronics

Each Stigmergic Block contains a central circuit board and six face circuit boards (Fig. 2). An ATMega328P microcontroller for running a block's software is mounted on the central circuit board and enables computation, data storage (using the SRAM, EEPROM, or Flash), and communication (using the LEDs, NFC transceiver, or Xbee module). A single USB connection provides a power source for recharging a lithium-ion battery, an interface for reprogramming the microcontroller, and an interface for debugging a block's software.

A push button is located near the USB port to switch the system's power on and off. The system's power is provided via two regulators. The first regulator provides 3.3 V for the control circuitry, while the second regulator provides 5 V for the LEDs on the face circuit boards.

We provide a socket for an Xbee wireless module on the central circuit board. The purpose of this wireless module is primarily to enable remote debugging and monitoring of a block. However, it is also possible to realize block-to-block and block-to-robot communication using this module. The Xbee module connects to the microcontroller using an emulated serial port, which is implemented using the microcontroller's 16-bit timer.
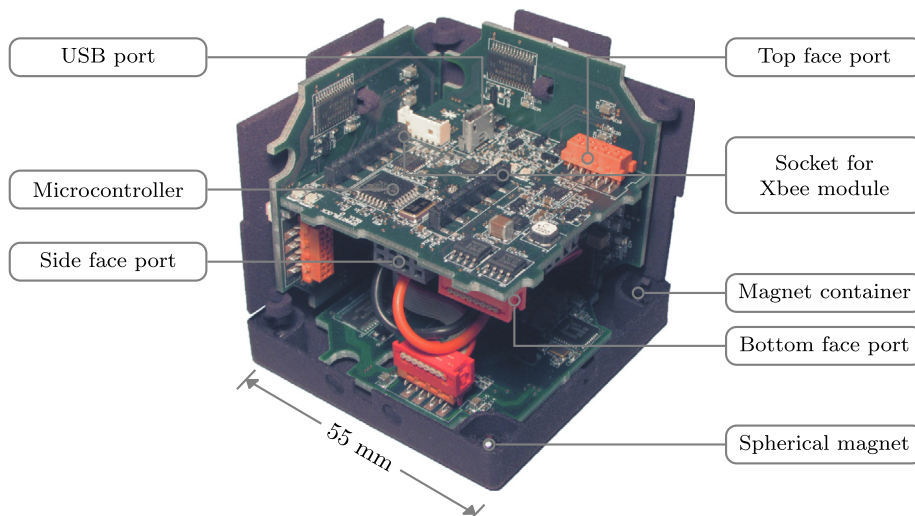
Fig. 2. Internal view of a Stigmergic Block.

Each face circuit board contains an NFC transceiver and an LED driver. The LED driver is used to set the brightness of the red, blue, and green channels of four multi-color LEDs. These LEDs can be detected by a BuilderBot's camera. The central circuit board provides six connectors for each of the face circuit boards. These connectors provide each face circuit board with power, an interrupt line, and an I^{2}C bus to control the LEDs and NFC transceiver.

2.1.2. Mechanical design

A Stigmergic Block is cubic in shape and has a side length of 55 mm. It is assembled from circuit boards, spherical magnets, and covers that we print using selective laser sintering. As shown in Fig. 3, we use three types of covers: a top cover, a side cover, and a bottom cover. We have designed the side covers to be used in an alternating up and down configuration. The top and bottom covers have their side cover slots and receptacles at different orientations to accommodate the up and down configuration of the side covers. The side covers contain printed springs, which have been orientated so that the adjacent side covers are held in place with tension. This configuration provides structural integrity while allowing the top and bottom covers to be easily removed.

The top and bottom covers both contain four small insets for four spherical neodymium magnets. These magnets are held in place using small tabs on the sides of each inset, which allow a magnet to be held in position while remaining free to rotate. These magnets cause adjacent blocks to self-align and increase the strength of a structure built from the blocks. The top cover contains a small hole located above the power and reset switch on the central circuit board. This hole allows the block to be turned on and off using a small screwdriver.
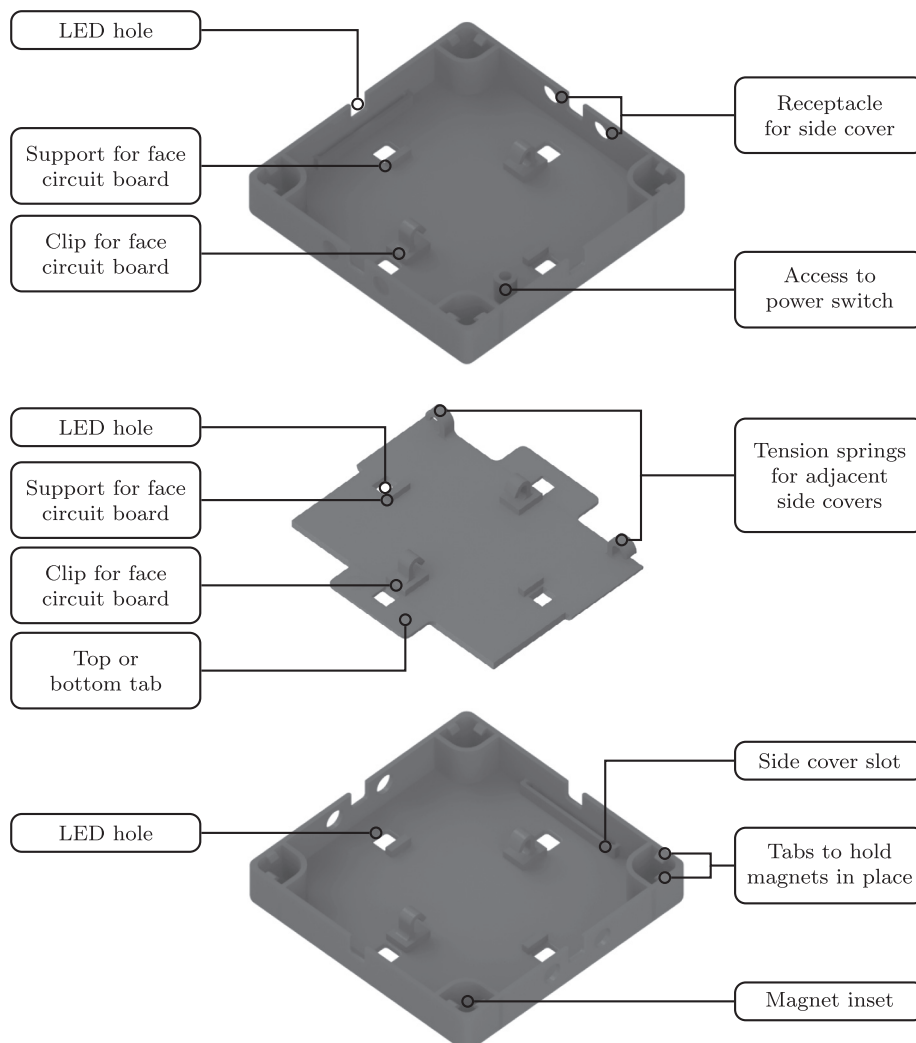

Fig. 3. Mechanical design of the top, side, and bottom covers of a Stigmergic Block.

2.1.3. Software

The microcontroller on the central circuit board contains 32 kilobytes of flash memory, 2 kilobytes of SRAM, and 1 kilobyte of EEPROM. The flash memory is partitioned to include the Optiboot bootloader, which enables the reprogramming of the microcontroller using the USB connection. The firmware for the Stigmergic Block is written in C++ and is derived from the Arduino AVR core library. The firmware starts by initializing the LED and NFC controllers, before entering an infinite loop where the block waits for an NFC message from a BuilderBot. When a message is received, it is used to configure the colors of the LEDs on a block.

At the time of writing, the firmware is relatively simple with respect to the potential functionality of a Stigmergic Block. To this end, we are currently investigating algorithms that use block-to-block communication to route messages through a structure. By enabling the routing of messages through the structure, we can investigate construction scenarios where the intelligence that coordinates the assembly process may be embedded not only in the robots but also in the building material itself.

2.2. BuilderBot

The BuilderBot evolved from our extension to the BeBot, a miniature mobile robotics platform [10]. The BuilderBot is powered by a dual-core processor from Texas Instruments which has access to 1 gigabyte of RAM and is clocked at 1 gigahertz. The BuilderBot moves around its environment using a differential drive and uses 12 rangefinders to avoid obstacles. LEDs in the base of the BuilderBot can be used for low-bandwidth communication with other robots and to provide visual feedback to human operators. An OmniVision camera is used to capture the scene in front of the robot.

A manipulator for assembling the Stigmergic Blocks into structures controls the vertical position of an end effector. This end effector is equipped with four semi-permanent electromagnets, which couple with the spherical magnets inside a Stigmergic Block to pick it up and to hold it in place during transport. By sending a pulse of current to the semi-permanent electromagnets, the strength of the magnetic field is either increased or decreased, improving alignment while picking up a block or detachment of a block respectively.

To locate the Stigmergic Blocks, the end effector is equipped with four rangefinders and a camera. We have mounted the camera at an angle of 45° from the horizontal. This angle is selected to provide a compromise between enabling the BuilderBot to track a Stigmergic Block at a distance and to track a block as it is being approached. Towards the end of the approach, a block can no longer be seen by the camera and the rangefinders on the end effector are used to perform the final alignment with the block prior to picking it up. The BuilderBot is capable of assembling structures up to three blocks in height.

While we have designed the BuilderBot for experiments in construction where the control software is autonomous, decentralized, and reactive, it is possible to connect the BuilderBots to a wireless network and to offload the required computer vision to a server. This server could also gather data from an overhead camera system and control the BuilderBots remotely in a centralized fashion.

2.2.1. Electronics

The BuilderBot electronics are implemented using six circuit boards: (i) the camera circuit board, (ii) the microprocessor circuit board, (iii) the interconnect circuit board, (iv) the power circuit board, (v) the manipulator circuit board, and (vi) the interface circuit board. The microprocessor circuit board contains a DuoVero COM, a single-board computer that runs Linux and that provides most of the BuilderBot's computational power. This single-board computer was selected due to its small size, its interfaces, and its processing power. However, since the processor on the DuoVero COM was released in 2011, there are now a number of alternative processors and single-board computer solutions that have comparable or better specifications. For example, the 3+ series of Raspberry Pi boards would make a strong candidate as an upgrade to the presented hardware. The number of circuit boards is predominantly a relic from the BeBot mobile robotics platform, to which a precursor of BuilderBot was implemented as an extension. If the presented hardware were to be upgraded, this upgrade would probably involve merging the functionality of the microprocessor circuit board, the power circuit board, and the interconnect circuit board onto a single circuit board that attaches to one of the Raspberry Pi 3+ boards. In the remainder of this section, we detail the purpose and the functionality of the circuit boards in the current design.

Camera circuit board: The BuilderBot includes a dedicated circuit board which supports an image sensor module from Leopard Imaging (Fig. 4). This image sensor module is based on the OV5640 image sensor from OmniVision. The camera circuit board provides a 24 MHz clock signal and a digital and analog power supply to the image sensor. We have mounted four white LEDs around the image sensor to increase the illumination of the scene. The image sensor is controlled over an I²C bus and its pixel data is routed to the microprocessor circuit board where it is captured by a dedicated image sensor interface.
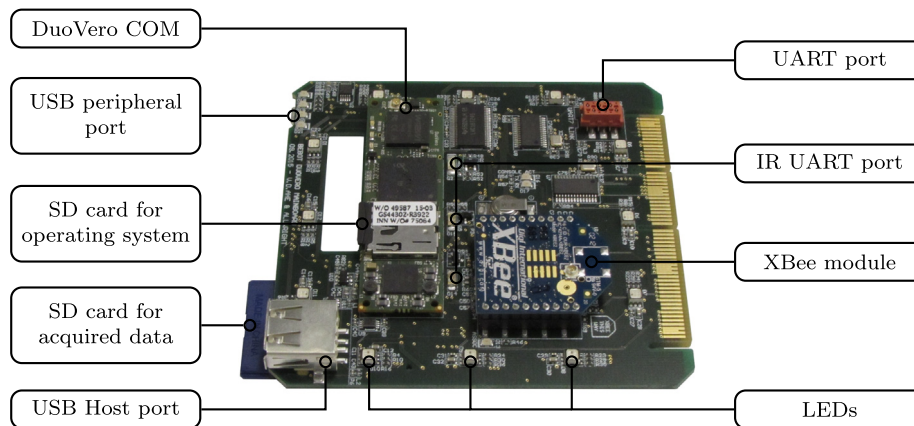
Microprocessor circuit board: Fig. 5 shows the DuoVero COM attached to the microprocessor circuit board. The DuoVero COM hosts the main microprocessor for the BuilderBot. This microprocessor is an OMAP4460 from Texas Instruments, which runs Linux. The DuoVero COM also provides WLAN and Bluetooth connectivity.

The microprocessor on the DuoVero COM includes two camera serial interface (CSI) ports, which can simultaneously capture video. We have routed both of these ports to two custom connectors, which can be connected to two of the aforementioned camera circuit boards. These connectors are located on the bottom of the microprocessor circuit board near the cut out on the left-hand side (see Fig. 5). Capturing images over CSI enables the use of the microprocessor's dedicated image processing hardware. This hardware can capture, scale, and compress the pixel data from a connected camera.
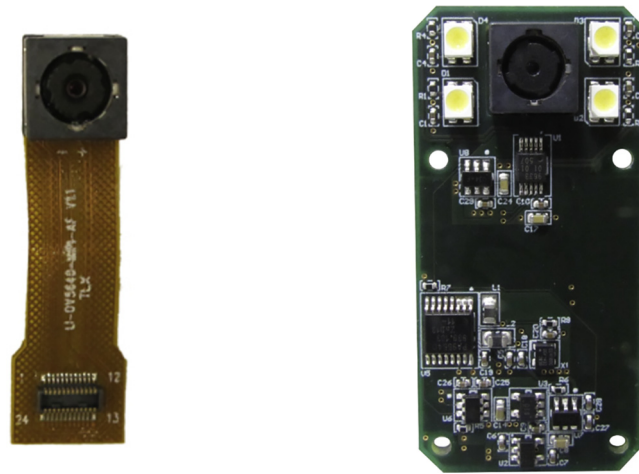
Fig. 4. Computer vision hardware for the BuilderBot robot. (left) A module from Leopard Imaging containing the OmniVision OV5640 image sensor. (right) A camera circuit board with an installed module.

**Fig. 5.** The microprocessor circuit board for the mobile robotics platform.

We have added an SD card reader to the microprocessor circuit board to store data such as the images captured by the BuilderBot's camera. A USB port is also provided and can be used to attach any standard USB device to the system.

The BuilderBot is designed to be connected to a PC via its micro-USB port. This port is routed to an integrated USB hub, which provides a user with low-level access to the system's bootloader (via an onboard USB-to-serial converter) and high-level access to the Linux operating system by emulating an Ethernet connection via USB On-The-Go (OTG). The integrated USB hub is compliant with the USB battery charging specification, enabling the BuilderBot to draw current from the USB connection in order to run its system or to charge its batteries. For debugging and low-bandwidth communication, twelve multi-color LEDs are evenly spaced around the perimeter of the microprocessor circuit board.

Interconnect circuit board: The primary function of the interconnect circuit board is to connect the microprocessor circuit board with the power circuit board. In addition to this interconnect, the interconnect circuit board contains three four-port I^2C multiplexers for connecting the rangefinders around the BuilderBot's chassis to the microprocessor circuit board.

Power circuit board: The power circuit board hosts two systems: the sensor-actuator system and the power management system (Fig. 6). Each system is controlled by an ATMega328P microcontroller. The sensor-actuator system provides differential drive to the BuilderBot by implementing closed-loop controllers for the left and right tracks. Shaft encoders in the motors enable the microcontroller to report changes in the position of the wheels to the high-level software running on the BuilderBot. The sensor-actuator system includes a digital gyroscope-accelerometer sensor.

The power management system is responsible for routing power and for recharging the BuilderBot's batteries. The power management is broken down into two domains: the system power domain and the actuator power domain. Both of these power domains have their own battery and power management integrated circuit (PMIC). External power can be applied to the system using either the standard 5.5/2.1 mm power jack on the power circuit board or the micro-USB connector on the microprocessor circuit board. The external power inputs are connected to the system PMIC, which forwards power
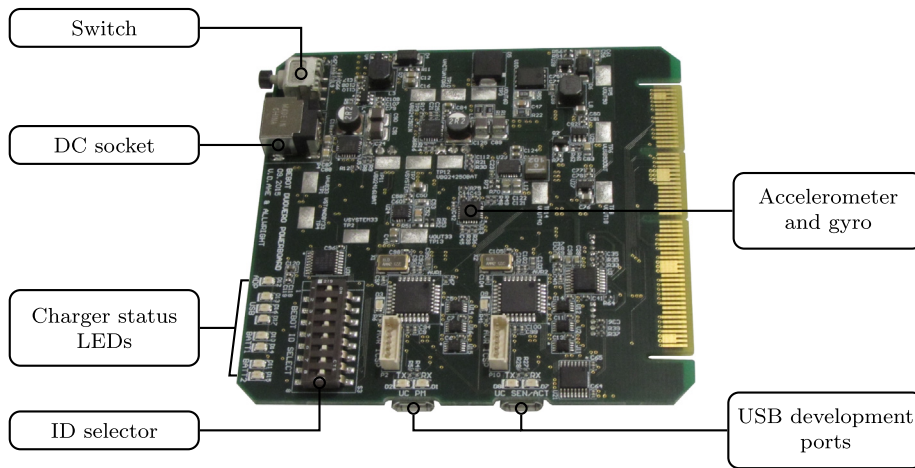
Fig. 6. The BuilderBot's power circuit board.

to the actuator PMIC when it is available. The firmware controlling the routing of power is implemented on the power management system's microcontroller. In addition to routing the available power, the firmware also configures the USB hub on the microprocessor circuit board and reads the result from the USB hub's battery charger detection circuitry to determine how much power can be sourced from the micro-USB connector.

Manipulator circuit board: The manipulator adjusts the height of an attached Stigmergic Block by raising and lowering its end effector. The height of the end effector is controlled using a stepper motor. Two limit switches are used for calibration and to keep the end effector inside of its operating range. Four semi-permanent electromagnets are located in the end effector, which couple with the spherical magnets in a Stigmergic Block. Depending on the direction of a current applied to the semi-permanent electromagnets, the magnetic field can be either strengthened or weakened. This current is generated by precharging four 6.8 millifarad capacitors to 25 V. The direction of the current is controlled using an H-bridge. The BuilderBot strengthens the magnetic field during block attachment. This strengthening of the field improves the alignment of the Stigmergic Block during attachment to the end effector. The BuilderBot weakens the magnetic field so that a block can detach from the end effector and become part of a structure.

The manipulator circuit board (Fig. 7) contains a microcontroller, which executes the manipulator's software and communicates with the BuilderBot's microprocessor circuit board via a universal asynchronous receiver-transmitter (UART). The manipulator circuit board has its own battery, which is charged independently using its USB port.

Interface circuit board: The interface circuit board (Fig. 8) contains an NFC transceiver for communicating with a Stigmergic Block. Two rangefinders are mounted directly on the interface circuit board, which the BuilderBot uses for aligning with a block or with a structure. The interface circuit board contains two connectors for two additional rangefinders which are con-
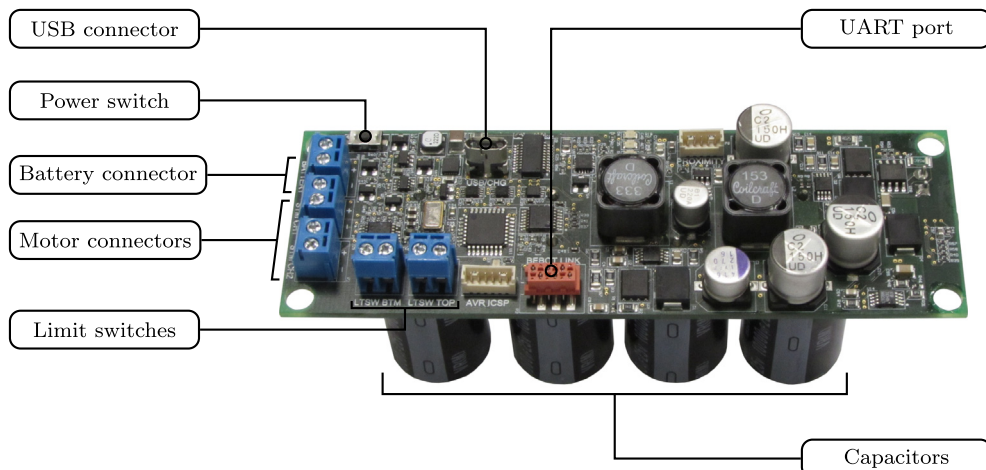
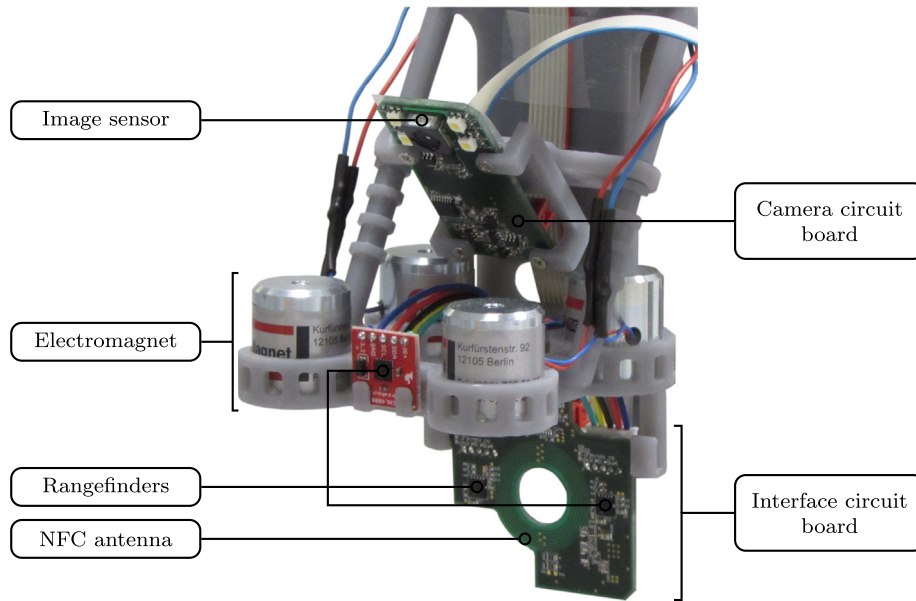**Fig. 7.** Manipulator circuit board.

Fig. 8. Interface circuit board attached to the manipulator.

nected to the end effector. The peripherals on the interface circuit board are controlled by the microcontroller on the manipulator circuit board.

2.2.2. Mechanical design

The BuilderBot contains two Faulhaber motor modules connected to a left and right track to form a differential drive, allowing the robot to move around its environment. These motor modules contain an inbuilt planetary gearbox and an encoder to provide feedback to the PID controller implemented on the sensor-actuator system's microcontroller. The motor modules, batteries, circuit boards, cables, and sensors are all attached to the BuilderBot's chassis which is assembled from 3D printed parts.

Following the diagram in Fig. 9, a stepper motor is used to drive the two lower sprockets via a pinion and worm gear. These sprockets are connected to chains that pull the end effector up and down. A counterweight is connected to each chain to offset the weight of the end effector and limit the load on the stepper motor. Limit switches are used to detect if the end effector is moving out of its operating range. The end effector contains four semi-permanent electromagnets which are used to pick up the Stigmergic Blocks. The BuilderBot is 38 cm tall and has a square footprint of 14 cm × 14 cm.

2.2.3. Software

We describe in this section the software for the BuilderBot. There are three layers to this software. The first layer is the firmware that runs on the BuilderBot's microcontrollers. The second layer involves the kernel modules that make the BuilderBot's sensors and actuators available to the Linux kernel. Finally, the third layer is the high-level software and libraries that enable an end user to write software that controls the BuilderBot's behavior.

Microcontroller firmware: Besides the microprocessor running the Linux operating system, the BuilderBot contains three microcontrollers. These microcontrollers communicate with the microprocessor using a command-based interface. The commands can set the speed or the position of a motor, request readings from remote sensors, and perform read/write operations from/to remote devices attached to a microcontroller's I²C bus. A valid command consists of a two-byte preamble, a type field that identifies the command, a length field that specifies the size of the data payload, the data payload, a checksum, and a two-byte postamble.

Fig. 10 shows an example command, which enables the actuator power supply on the BuilderBot.

Linux operating system and drivers: The BuilderBot microprocessor runs a custom variant of Linux which is built using the Yocto build system.[1] A configuration layer for the BuilderBot (called meta-builderbot) is provided with the project and can be used to quickly build and customize the operating system. The Yocto build system describes a target system in terms of configuration layers. These configuration layers contain scripts called recipes, which instruct the build system to fetch, patch, configure, compile, and install software for the target system.

---

[1] Yocto Project:www.yoctoproject.org/.

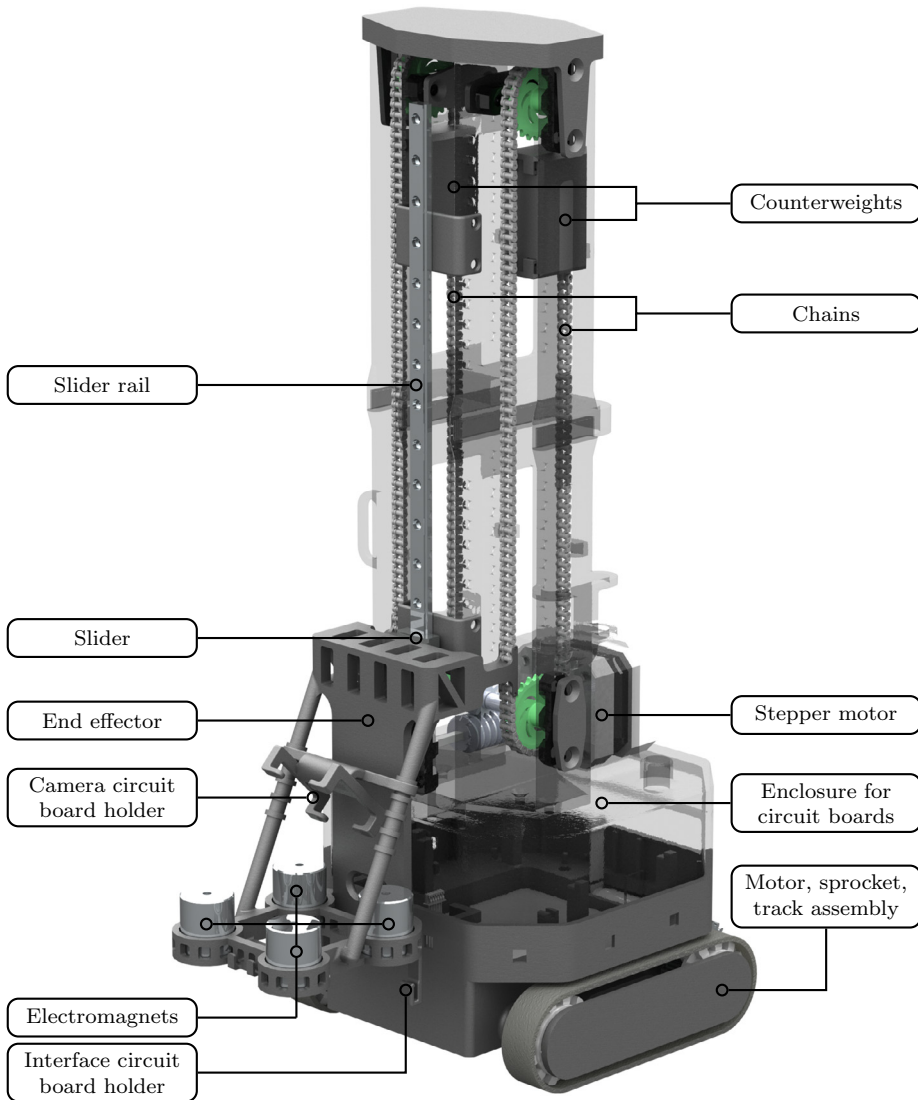**Fig. 9.** Mechanical components of the BuilderBot.

F0	CA	40	01	01	42	53	0F
Preamble		Type	Length	Data	Checksum	Postamble	

Fig. 10. A data packet for enabling the actuator power supply using the command-based interface.

The interfaces to the microcontroller are implemented as external, loadable kernel modules. These kernel modules are instantiated and configured using a device tree which is included in the Linux kernel recipe. From the perspective of the Linux operating system, each microcontroller is represented as a multi-function device which is connected to the processor using a serial device bus. Each multi-function device hosts several child devices which send commands over the serial device bus and are executed by the microcontroller's firmware. These child devices are all implemented using the standard subsystems available to the Linux operating system. For example, the drivers for the BuilderBot's sensors and actuators are realized using the Industrial I/O (IIO) subsystem and video capture is supported using the Video for Linux 2 (V4L2) subsystem.

Several patches in the meta-builderbot layer have been applied to the Linux kernel to support upstream features that have not yet been merged into the Linux mainline. For example, we have patched the kernel to include support for low-

Table 1
Design file summary for the Stigmergic Block.

Design file name	File type	License	Location (OSF Link)
Electronics/FaceBoard.zip	Altium Designer Project	MIT	https://osf.io/vqd84
The project for the face circuit board. This circuit board contains the LEDs and NFC transceivers for the Stigmergic Block.			
Electronics/MainBoard.zip	Altium Designer Project	MIT	https://osf.io/q7khx
The project for the main circuit board. The main circuit board provides power management and contains the microcontroller that runs a Stigmergic Block's software.			
Mechanical Design/BottomFace.FCStd	FreeCAD Part	MIT	https://osf.io/7gqxw
The 3D model of the bottom cover of the Stigmergic Block.			
Mechanical Design/SideFace.FCStd	FreeCAD Part	MIT	https://osf.io/qxhmf
The 3D model of the side cover of the Stigmergic Block.			
Mechanical Design/TopFace.FCStd	FreeCAD Part	MIT	https://osf.io/bndx5
The 3D model of the top cover of the Stigmergic Block.			
Software/Firmware.tar.gz	C++ Source Code	LGPL	https://osf.io/y5tj4
The bootloader and firmware for the Stigmergic Block (based on the Arduino core library for AVR microcontrollers).			

speed IIO output buffers, which are used to control the BuilderBot's actuators. Without these patches, some of the Builder-Bot's drivers will not compile.

Userspace libraries and applications: The system image for the BuilderBot comes with three libraries for (i) detecting tags, (ii) interfacing with the IIO subsystem, and (iii) interfacing with the V4L2 subsystem. Yavta, a userspace application for testing the V4L2 subsystem, is also included and can be used for capturing frames from the BuilderBot's camera.

To control the behavior of the BuilderBot, we use an executable based on the libraries of ARGoS, a multi-robot simulator [16]. This executable initializes the BuilderBot's sensors and actuators and provides a high-level control interface, which enables a user to implement the control software for the BuilderBot using the Lua scripting language or C++. This control interface is provided by the ARGoS libraries and is an abstraction layer on top of the physical hardware. In future work, we intend to use this abstraction layer to create a simulation model of the BuilderBot that is compatible with the ARGoS simulator and that enables the running of experiments with large numbers of robots and blocks inside the ARGoS simulator using the same control software as the hardware.

3. Design files

The following tables in this section provide a summary of the design files required to manufacture the presented hardware. The 3D printed parts were designed using FreeCAD 0.17-7 and the circuit boards were designed using Altium Designer 14. The interconnect circuit board was developed more recently than the other circuit boards and was designed in KiCad 5.0, an open source EDA.

For each circuit board, we provide an archive that contains the complete project in addition to a PDF document for use as a quick reference for checking the functionality and layout of the circuit boards. For the 3D printed parts, we provide a FreeCAD part document for editing purposes and an STL file for submission to a rapid prototyping facility.

The design files for the Stigmergic Block are summarized in Table 1 and the design files for the BuilderBot are summarized in Table 2.

4. Bill of materials

The bill of materials for this project is split into two spreadsheets which are located in the project's repository on the Open Science Foundation's website. Due to the amount of information involved in detailing the individual components for each of the eight circuit boards, we have decided to list each circuit board as a single item on the bill of materials. The cost of these items includes manufacturing the boards, ordering the components, and having the boards assembled. To this end, a user of the project repository needs to generate a detailed bill of materials for the individual circuit boards using either Altium Designer or KiCad. This solution is practical since fabricators and board assemblers may use their own spreadsheets which can be automatically populated using EDA software. The following links are to the online spreadsheets for the Stigmergic Block and for the BuilderBot.

File	Format	OSF Link
Bill of materials for the Stigmergic Block	Open Document Spreadsheet	https://osf.io/pzhgn/
Bill of materials for the BuilderBot	Open Document Spreadsheet	https://osf.io/svb58/

Table 2
Design file summary for the BuilderBot

Design file name	File type	License	Location (OSF Link)
Electronics/CameraBoard.zip The project for the camera circuit board, which provides power and control signals for the OV5640 image sensor module.	Altium Designer Project	MIT	https://osf.io/hys5p
Electronics/InterconnectBoard.zip The project for the interconnect circuit board, which connects the power circuit board to the microprocessor circuit board. This board also connects the microprocessor board to the rangefinders around the chassis.	KiCad Project	MIT	https://osf.io/3me78
Electronics/InterfaceBoard.zip The project for the interface circuit board, which contains the NFC transceiver that communicates with the Stigmergic Blocks.	Altium Designer Project	MIT	https://osf.io/pyt37
Electronics/ManipulatorBoard.zip The project for the manipulator circuit board, which controls the lift actuator system.	Altium Designer Project	MIT	https://osf.io/8sb4j
Electronics/MicroprocessorBoard.zip The project for the microprocessor circuit board, which hosts the DuoVero COM. The DuoVero COM is a single board computer that runs Linux.	Altium Designer Project	MIT	https://osf.io/vm7kx
Electronics/PowerBoard.zip The project for the power circuit board, which manages the BuilderBot's system and actuator power supplies.	Altium Designer Project	MIT	https://osf.io/b6nj5
Mechanical Design/BaseLowerChassis.FCStd The 3D model for the base lower chassis, which supports the motors and has a cut out for the system battery.	FreeCAD Part	MIT	https://osf.io/n3zch
Mechanical Design/BaseShaftCoupler.FCStd The 3D model for the base shaft coupler, which connects the stepper motor to the shafts that move the manipulator.	FreeCAD Part	MIT	https://osf.io/rxm2s
Mechanical Design/BaseShaftSpacer.FCStd The 3D model for the base shaft spacer, which maintains the correct distance between the gears, the pinion, and the worm gear.	FreeCAD Part	MIT	https://osf.io/r7k28
Mechanical Design/BaseTrackHolder.FCStd The 3D model for the base track holder, which holds the sprockets and tracks for the differential drive in place.	FreeCAD Part	MIT	https://osf.io/w25eu
Mechanical Design/BaseUpperChassis.FCStd The 3D model for the base upper chassis, which contains the supports for the interconnect, power, and microprocessor circuit boards.	FreeCAD Part	MIT	https://osf.io/v36ra
Mechanical Design/CameraChassis.FCStd The 3D model for the camera chassis, which facilitates the attachment of the camera circuit board to the lift actuator's manipulator.	FreeCAD Part	MIT	https://osf.io/asvez
Mechanical Design/CounterweightCover.FCStd The 3D model for the counterweight covers, which keep the lead counterweights in place and which connect to the chains.	FreeCAD Part	MIT	https://osf.io/5zxsp
Mechanical Design/CounterweightHolder.FCStd The 3D model for the counterweight holders, which hold the lead counterweights and which are connected to the counterweight covers.	FreeCAD Part	MIT	https://osf.io/rupmv
Mechanical Design/LiftActuatorBase.FCStd The 3D model for the lift actuator base, which encloses the base of the robot and which provides a platform to which the lift actuator columns are attached.	FreeCAD Part	MIT	https://osf.io/rjxqd
Mechanical Design/LiftActuatorBracket.FCStd The 3D model for the lift actuator bracket, which fixes the slider rail to the lift actuator columns.	FreeCAD Part	MIT	https://osf.io/xh5kp
Mechanical Design/LiftActuatorColumnLowerLeft.FCStd The 3D model for the lift actuator lower left column. This part contains a support for the lower limit switch.	FreeCAD Part	MIT	https://osf.io/wxad9
Mechanical Design/LiftActuatorColumnLowerRight.FCStd The 3D model for the lift actuator lower right column. This part contains the battery holder for the manipulator circuit board.	FreeCAD Part	MIT	https://osf.io/34w6m
Mechanical Design/LiftActuatorColumnUpperLeft.FCStd The 3D model for the lift actuator upper left column. This part contains a support for the upper limit switch.	FreeCAD Part	MIT	https://osf.io/vrzah
Mechanical Design/LiftActuatorColumnUpperRight.FCStd The 3D model for the lift actuator upper right column.	FreeCAD Part	MIT	https://osf.io/4n2em
Mechanical Design/LiftActuatorCover.FCStd The 3D model for the lift actuator cover, which keeps the two upper lift actuator columns aligned with each other.	FreeCAD Part	MIT	https://osf.io/ywg8f
Mechanical Design/LiftActuatorManipulator.FCStd The 3D model for the lift actuator manipulator, which enables the robot to interact with the Stigmergic Blocks. This part contains supports for the camera module, the electromagnets, and the interface circuit board.	FreeCAD Part	MIT	https://osf.io/8t349

Table 2 (*continued*)

Design file name	File type	License	Location (OSF Link)
Mechanical Design/LiftActuatorShaftCover.FCStd	FreeCAD Part	MIT	https://osf.io/cfsxp
The 3D model for the lift actuator shaft covers, which hold the lift actuator's shafts in place.			
Software/SystemImage.tar.gz	Yocto Layer	MIT	https://osf.io/s79wh
This archive contains a layer for the Yocto build system for the BuilderBot.			
Software/Firmware.tar.gz	C++ Source Code	LGPL	https://osf.io/k6qh3
This archive contains the bootloader and firmware for the BuilderBot's microcontrollers. The code is based on Arduino's core libraries for AVR microcontrollers.			

Table 3
GitHub Repositories for software components.

Software component	OSF Link	GitHub Repository
Stigmergic Block firmware and bootloader	https://osf.io/y5tj4	https://github.com/allsey87/stigmergic-block-firmware
BuilderBot firmware and bootloader	https://osf.io/k6qh3	https://github.com/allsey87/builderbot-firmware
Yocto system image layer for the BuilderBot	https://osf.io/s79wh	https://github.com/allsey87/meta-builderbot
BuilderBot plug-in for ARGoS 3	None – built automatically as part of the system image	https://github.com/allsey87/argos3-builderbot

5. Build instructions

Due to the large number of steps required to assemble the hardware, we have opted to present this content in the form of two detailed videos that describe the assembly process in its entirety. The videos are located in the project's repository on the Open Science Foundation's website. Please use the following links to download and to view these videos.

File	Format	OSF Link
Assembly instructions for the Stigmergic Block	MP4 (H.264)	https://osf.io/kb3g6/
Assembly instructions for the BuilderBot	MP4 (H.264)	https://osf.io/8bkjn/

6. Operation instructions

In this section, we discuss how to operate the hardware. At the time of writing, snapshots of the firmware, configuration files, and high-level software have been stored in the project's repository on the Open Science Foundation's website. These snapshots are the versions of the software referred to in this article. More recent versions of this software may be found in the GitHub repositories listed in Table 3. The versions from these repositories, however, may be in development or may contain changes that are incompatible with the other software components.

For readers who are interested in replicating our hardware, Section 6.1 and Section 6.2 describe how to download, compile, install, and interact with the bootloaders and the firmware in our system and how to prepare a system image for the BuilderBot respectively. For more general readers, we recommend skipping forward to Section 6.3 where we demonstrate how to run the example routines.

6.1. Installing the bootloaders and firmware

The circuit boards in the presented hardware are controlled using ATMega328P microcontrollers. There is a single microcontroller in the Stigmergic Block and three microcontrollers in the BuilderBot. The bootloader and firmware for each of these microcontrollers is provided as a gzipped tarball that is located in the project's repository on the Open Science Foundation's website. The archives contain a folder for the bootloader and a folder for the firmware for each microcontroller. The following instructions assume that you are using Linux and that you have installed avrdude and the AVR versions of the binutils, gcc, and libc packages. In this section, we describe the process of installing the bootloader and firmware on a Stigmergic Block. This process, however, is also applicable to the three microcontrollers in the BuilderBot.

Before the firmware for a microcontroller can be downloaded over the USB port, the Optiboot bootloader[2] must first be installed. There are two approaches to installing the bootloader. The bootloader can either be given to a component distributor (for example, Digi-Key), who will program the microcontrollers prior to shipping them or the bootloader can be manually installed to the microcontroller using avrdude and a Bus Pirate (see Fig. 11). In the remainder of this section, we will describe the latter approach.

---

[2] Optiboot: https://github.com/optiboot/optiboot.
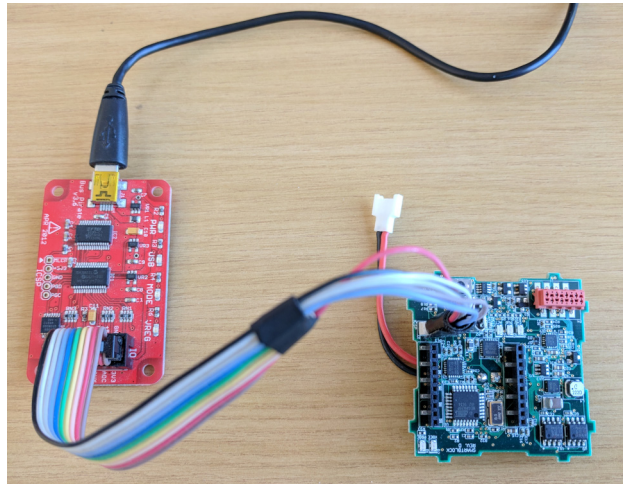
Fig. 11. Installing the bootloader to the Stigmergic Block using avrdude and a Bus Pirate. A custom cable connects the Bus Pirate to the ICSP (in-circuit serial programmer) port.

The bootloader/firmware archive for the Stigmergic Block can be downloaded, extracted, compiled, and installed by issuing the following commands. We note at this point that in order to improve the reliability of the serial connection, we have used a baud rate of 57 600 throughout our code base. In the following commands, references to $\texttt{ttyUSBX}$ must be replaced by the tty that represents a Bus Pirate connected to the ICSP (in-circuit serial programmer) port on the main circuit board.

```
# create a new directory for working with the bootloader
mkdir workspace && cd workspace
# download and extract the bootloader/firmware archive
curl -L https://osf.io/y5tj4/download | tar xz
# compile the bootloader
make -C block/bootloader
# download the bootloader (change ttyUSBX to the tty for the bus pirate)
avrdude -c buspirate -p m328p -P /dev/ttyUSBX -U lock:w:0x3F:m -U lfuse:w:0xFF:m \
-U hfuse:w:0xDE:m -U efuse:w:0x05:m -U flash:w:block/bootloader/optiboot_atmega328.hex \
-U lock:w:0x0F:m
```

The parameters in the avrdude command have the following effects: (i) the flash memory for the bootloader is unlocked, (ii) the microcontroller is configured to use an external clock and the size of its bootloader is set, (iii) the bootloader is uploaded, and (iv) the flash memory for the bootloader is locked. After the bootloader has been installed, it is possible to program the microcontroller and to communicate with it using the USB port – the Bus Pirate is no longer required. The following commands result in the firmware for the Stigmergic Block being compiled and installed over USB. These commands assume that the current directory is where the previous commands were issued.

```
# compile the firmware for the Stigmergic Block
make -C block/firmware
# upload the firmware to the Stigmergic Block
# (change ttyUSBX to the tty for the Stigmergic Block's USB connection)
avrdude -c arduino -p m328p -P /dev/ttyUSBX -b 57600 \
-U flash:w:block/firmware/build/firmware.hex
```

At the time of writing, the commands for the BuilderBot's microcontrollers consist of a sequence of bytes that include a preamble, a data payload, a checksum, and a postamble. This interface is necessary to efficiently move multi-byte data between the microcontrollers and the Linux operating system running on the BuilderBot.

Table 4
Character-based commands for the Stigmergic Block.

Command	Expected response/behavior
a	Reads the accelerometer and prints its output to the terminal.
b	Reads the battery level and prints its value to the terminal.
l	Tests the LEDs by cycling through a pattern of colors.
p	Prints the power status, that is, whether external power is being supplied and whether the battery is charging.
r	Performs a hard reset of the microcontroller.
t	Sends an NFC message from the first connected port, where the order of the ports is: north, east, south, west, top, bottom.
u	Prints the up time of the microcontroller, that is, the number of miliseconds that the firmware has been running.
0	Turns off the LEDs.
1–4	Sets the LEDs to one of four predefined colors which are the corners of (Y) UV color space.

The microcontroller on the Stigmergic Block, however, uses a character-based interface and can be manipulated by issuing single character commands with a terminal emulation program such as putty, minicom, or picocom. When connecting from these programs to the Stigmergic Block, keep in mind that the baud rate is 57 600. Table 4 summarizes the character-based commands for the Stigmergic Block. We note that if an Xbee wireless module is installed inside the block, the input and output will be automatically redirected over this connection.

6.2. Preparing the system image

The system image is prepared using the Yocto build system. In order to use the Yocto build system, you will need to install the packages listed in the Yocto Project Quick Build Manual.[3]

Building the system image involves compiling and preparing the Linux operating system, the drivers for the BuilderBot, and its userspace tools and software. While the entire process is automatic, it will require approximately 30 gigabytes of hard disk space and will take between 2 and 8 h to complete depending on the speed of the CPU and the speed of the internet connection. Issuing the following commands will result in the Yocto layers being cloned from the project's server, the layer for the BuilderBot being downloaded and extracted, and the build process being started.

```
# clone the Yocto project repository and create a new branch based on tags/yocto-2.5
git clone git://git.yoctoproject.org/poky.git
cd poky
git checkout tags/yocto-2.5 -b my-yocto-2.5
# download and extract the system image layer for the BuilderBot
curl -L https://osf.io/s79wh/download | tar xz # initialize the build directory
# initialize the build directory
# this command will change the current directory to the build directory
TEMPLATECONF=meta-builderbot/conf source oe-init-build-env
# start the build process for the system image
bitbake console-image-builderbot
```

Once the system image has been built, it needs to be loaded onto a microSD card that can be booted by the DuoVero COM. To prepare this microSD card, follow the instructions from the Gumstix website for creating a bootable microSD card.[4]

At this point, the BuilderBot can be switched on. First, insert the microSD card that you have prepared into the DuoVero COM. Second, attach a microUSB cable between the microprocessor circuit board and the PC that you are using for development. Connect both batteries and press the power button (the upper push button on the power circuit board). Once the system power is switched on, a ttyUSBX device will appear on the development PC, which enables interaction with the U-Boot bootloader and with the Linux operating system. By default, the system is configured with only the root user and with no password set. Once logged in, standard Linux tools and configuration files can be used to set up networking using WLAN or USB On-The-Go.

6.3. Running the example routines

The BuilderBot is controlled using an executable called argos3 which links against the core libraries of the ARGoS simulator at run-time. The argos3 executable is responsible for initializing the BuilderBot's sensors and actuators and for

---

[3] Yocto Project Quick Build Manual:https://www.yoctoproject.org/docs/2.5/brief-yoctoprojectqs/brief-yoctoprojectqs.html.
[4] https://www.gumstix.com/support/getting-started/create-bootable-microsd-card/.

executing the functions provided in a Lua script, which defines a BuilderBot's behavior. The Lua script is specified in the configuration file that is passed to the argos3 executable on the command line. We have provided four example routines that are deployed automatically in the home directory of the root user.

Routine	Command	Expected Behavior
Differential drive	argos3 -c test_differential_drive.argos	The BuilderBot will turn on the spot with increasing velocity.
Electromagnets	argos3 -c test_electromagnet_system.argos	Once the capacitors on the manipulator circuit board are charged, the destructive current will be routed to the electromagnets causing a Stigmergic Block to fall free.
Lift System	argos3 -c test_lift_system.argos	The lift actuator will perform a self-calibration routine before moving the actuator from its bottom position to its top position and back.
Rangefinders	argos3 -c test_rangefinders.argos	This routine reads the values of all the rangefinders and prints them to the console.

7. Validation and characterization

In this section, we specify the capabilities of the presented hardware. For high-level validation purposes, we have prepared a video that demonstrates the key functionality of the hardware. In addition to this video, the following subsections list the capabilities of the hardware and provide its specifications.

File	Format	OSF Link
System verification video	MP4 (H.264)	https://osf.io/2ujc4/

7.1. Stigmergic Block

The Stigmergic Block is capable of communicating via its LEDs, its NFC transceiver, and an optional Xbee module. Each red, blue, and green channel of each LED on the Stigmergic Block can be set independently with 8-bits of resolution between off and fully on. The range of the NFC transceiver is approximately 2 cm, which prevents a Stigmergic Block from communicating with itself while achieving a reliable connection with adjacent blocks or a BuilderBot. The range of the Xbee link depends on which variant of the module is used. With the PCB antenna variant of the Xbee, we have achieved a range of approximately 0.5 m.

The spherical magnets in the two blocks are strong enough to bring two blocks into alignment when they are separated by up to 1 cm or misaligned by up to 25°.

Although dependent on usage (for example, the frequency of communication and LED brightness) the average running time for the block is approximately 7 h with a 500 milliampere-hour lithium-ion battery. The weight of the block is 110 grams. The side length of the cube is 55 mm.

7.2. BuilderBot

The BuilderBot has a height of 38.8 cm and a square footprint with a side length of 14 cm. The weight of the BuilderBot is 2.1 kilograms.

When the BuilderBot's lift actuator is positioned at its maximum height from the ground (3.5 blocks, or 19.25 cm), the camera can see the blocks on the ground up to approximately 35 cm away from the center of the robot. When the lift actuator's manipulator is positioned at its minimum height from the ground (1 block, or 5.5 cm) a block can be detected until just before it is attached to the manipulator. It takes approximately 5 s to pre-charge the capacitors prior to attaching a block to the lift actuator's manipulator or detaching it from the manipulator. However, since this pre-charging can occur as the robot approaches a block or a structure, this delay is mostly negligible in practice. Once the block is attached to the manipulator, the semi-permanent electromagnets are strong enough to hold a single block without applying an electrical current. The vertical speed of the lift actuator is 1.6 cm per second.

With the current tuning of the differential drive system, we drive the robot forward with a velocity of 3.5 cm per second and can have the robot turn on the spot with an angular velocity of 35° per second.

The BuilderBot is capable of communicating over WLAN, Bluetooth, NFC (with a Stigmergic Block), Zigbee (via an optional Xbee module), and using its LEDs (with other robots via their cameras). The battery configuration of the BuilderBot provides a running time of approximately 5 h.

When running the AprilTag algorithm on the robot with good illumination, it is possible to process and detect tags on board the DuoVero COM at a rate of 5 frames per second with an input resolution from the OV5640 camera module of 640×360 pixels.

8. Conclusions

We have presented a completely open source robotics construction system for performing experiments in multi-robot construction in laboratory settings. Our system consists of a building material called the Stigmergic Blocks and robots called BuilderBots. The robots have been designed to assemble the Stigmergic Blocks into structures so that we can study multi-robot construction. The system has been designed to facilitate experiments where the intelligence that coordinates the construction can be embedded not only in the robots but also in the building material.

In future work, we plan to extend the hardware to address some of the current limitations with the manipulator. In particular, we are currently studying new manipulator designs that will allow the BuilderBot to both add and remove the Stigmergic Blocks from a structure. This capability will enable us to study reconfigurable structures that adapt to the environment over time. Due to the associated costs of running experiments, we are also currently working on a simulation model of both the BuilderBot and Stigmergic Block so that we can run experiments inside of the ARGoS simulator. These simulation models will be based on the plug-ins presented in [2].

This article is designed to complement the project's repository, which is hosted on the Open Science Foundation's website [3]. For those who are interested in implementing the presented hardware, we would strongly encourage contacting the corresponding author to discuss potential collaboration.

Conflict of interest

The authors declare that they have no conflicts of interest in the presented work.

Acknowledgements

References

[1] M. Allwright, N. Bhalla, M. Dorigo, Structure and markings as stimuli for autonomous construction, Proceedings of the Eighteenth International Conference on Advanced Robotics, IEEE, 2017, pp. 296–302, https://doi.org/10.1109/icar.2017.8023623.

[2] M. Allwright, N. Bhalla, C. Pinciroli, M. Dorigo, Simulating multi-robot construction in ARGoS, Proceedings of the Eleventh International Conference on Swarm Intelligence, Springer, 2018, pp. 188–200, https://doi.org/10.1007/978-3-030-00533-7_15.

[3] M. Allwright, W. Zhu, M. Dorigo, An open-source multi-robot construction system, 2018.https://doi.org/10.17605/osf.io/spfx7.

[4] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M.W. Mueller, J.S. Willmann, F. Gramazio, M. Kohler, R. D'Andrea, The flight assembled architecture installation: cooperative construction with flying machines, IEEE Control Syst. 34 (4) (2014) 46–64, https://doi.org/10.1109/mcs.2014.2320359.

[5] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, 1999.

[6] O.H. Bruinsma, An Analysis of Building Behaviour of the Termite Macrotermes subhyalinus (Rambur) (Ph.D. thesis), Wageningen University, Netherlands, 1979.

[7] S. Camazine, J.L. Deneubourg, N.R. Franks, J. Sneyd, G. Theraulaz, E. Bonabeau, Self-Organization in Biological Systems, Princeton University Press, 2001.

[8] M. Dorigo, M. Birattari, M. Brambilla, Swarm robotics, Scholarpedia 9 (1) (2014) 1463, https://doi.org/10.4249/scholarpedia.1463.

[9] P.P. Grassé, Reconstruction of the nest and coordination between individuals in terms. Bellicositermes Natalensis and Cubitermes sp. the theory of stigmergy: test interpretation of termite constructions, Insectes Sociaux 6 (1) (1959) 41–80, https://doi.org/10.1007/bf02223791.

[10] S. Herbrechtsmeier, U. Witkowski, U. Rückert, BeBot: A modular mobile miniature robot platform supporting hardware reconfiguration and multi-standard communication, Progress in Robotics, Springer, 2009, pp. 346–356, https://doi.org/10.1007/978-3-642-03986-7_40.

[11] C. Jones, M.J. Matarić, Automatic synthesis of communication-based coordinated multi-robot systems, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2004, pp. 381–387, https://doi.org/10.1109/iros.2004.1389382.

[12] I. Karsai, Z. Pénzes, Comb building in social wasps: self-organization and stigmergic script, J. Theor. Biol. 161 (4) (1993) 505–525, https://doi.org/10.1006/jtbi.1993.1070.

[13] Q. Lindsey, D. Mellinger, V. Kumar, Construction with quadrotor teams, Auton. Robots 33 (3) (2012) 323–336, https://doi.org/10.1007/s10514-012-9305-0.

[14] E. Olson, AprilTag: a robust and flexible visual fiducial system, 2011 IEEE International Conference on Robotics and Automation, IEEE, 2011, pp. 3400–3407, https://doi.org/10.1109/icra.2011.5979561.

[15] K. Petersen, R. Nagpal, J. Werfel, TERMES: an autonomous robotic system for three-dimensional collective construction, Proceedings of Robotics: Science and Systems, RSS Foundation, 2011, pp. 257–264, https://doi.org/10.15607/rss.2011.vii.035.

[16] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L.M. Gambardella, M. Dorigo, ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems, Swarm Intelligence 6 (4) (2012) 271–295, https://doi.org/10.1007/s11721-012-0072-5.

[17] C. Sugawara, Y. Doi, Collective construction of dynamic equilibrium structure through interaction of simple robots with semi-active blocks, Distributed Autonomous Robotic Systems 12, Springer, 2016, pp. 165–176, https://doi.org/10.1007/978-4-431-55879-8_12.

[18] G. Theraulaz, E. Bonabeau, A brief history of stigmergy, Artif. Life 5 (2) (1999) 97–116, https://doi.org/10.1162/106454699568700.

[19] J. Werfel, K. Petersen, R. Nagpal, Designing collective behavior in a termite-inspired robot construction team, Science 343 (6172) (2014) 754–758, https://doi.org/10.1126/science.1245842.

[20] J. Worcester, M. Ani Hsieh, R. Lakaemper, Distributed assembly with online workload balancing and visual error detection and correction, Int. J. Robot. Res. 33 (4) (2014) 534–546, https://doi.org/10.1177/0278364913509125.