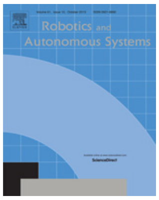
Robotics and Autonomous Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Kilobot: A low cost robot with scalable operations designed for collective behaviors

Michael Rubenstein [a,*], Christian Ahler [a], Nick Hoff [a], Adrian Cabrera [b,a], Radhika Nagpal [a]

[a] Harvard University, United States
[b] EPFL, Switzerland

HIGHLIGHTS

- Present design of Kilobot, a low cost robot for testing collective algorithms on large groups.
- Easy to use with a single user.
- Demonstrations of collective algorithms on up to 100 robots.

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Modular robot
Multi-robot system
Robot collectives

ABSTRACT

In current robotics research there is a vast body of work on algorithms and control methods for groups of decentralized cooperating robots, called a swarm or collective. These algorithms are generally meant to control collectives of hundreds or even thousands of robots; however, for reasons of cost, time, or complexity, they are generally validated in simulation only, or on a group of a few tens of robots. To address this issue, this paper presents Kilobot, an open-source, low cost robot designed to make testing collective algorithms on hundreds or thousands of robots accessible to robotics researchers. To enable the possibility of large Kilobot collectives where the number of robots is an order of magnitude larger than the largest that exist today, each robot is made with only $14 worth of parts and takes 5 min to assemble. Furthermore, the robot design allows a single user to easily operate a large Kilobot collective, such as programming, powering on, and charging all robots, which would be difficult or impossible to do with many existing robotic systems. We demonstrate the capabilities of the Kilobot as a collective robot, by using a small robot test collective to implement four popular swarm behaviors: foraging, formation control, phototaxis, and synchronization.

1. Introduction

A large group of decentralized closely cooperating entities, commonly called a collective or swarm, can work together to complete a task that is beyond the capabilities of any of its individuals. Many such examples can be found in nature: army ants and honeybee colonies effectively forage over large areas many kilometers wide; desert ant groups can collectively transport large irregular objects 50 times their collective weight; termite colonies construct mounds meters tall even though the individuals are only a few millimeters tall themselves. These examples from nature have inspired long-standing research in collective robotics to achieve the kind of parallelism, robustness and collective capability of these natural systems.

Within robotics, there is a wide range of active research topics that explore algorithms to control these robotic collectives, such as self-assembly [1–3], collective construction [4,5], and exploration [6,7], to name a few. Researchers commonly envision these algorithms to operate on collectives of hundreds [4], thousands [2,3], or more [1,8], robots; however, for reasons of cost, time, or complexity, they are generally validated in simulation only [1,3], or on a group of a few tens of robots or fewer [9,10]. When using a simulation to validate an algorithm for a collective of robots, it is difficult to accurately model robots' interaction with each other, such as communication and sensing, and with the environment, such as movement and collisions. This modeling difficulty can lead to disparities in algorithm behavior when operating on a simulated collective versus a real robotic collective. Additionally, operating an algorithm designed for a large collective of robots on just a few may hide scaling issues within the algorithm that can only be uncovered in a much larger collective. To better understand and validate both current and future collective control algorithms, it would be useful for these algorithms to be tested on a larger collective of real robots.

Other research groups have also recognized the importance of a collective of robots for testing and validating algorithms;

---

[*] Corresponding author.
E-mail address: mrubenst@seas.harvard.edu (M. Rubenstein).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.08.006

Please cite this article in press as: M. Rubenstein, et al., Kilobot: A low cost robot with scalable operations designed for collective behaviors, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.08.006

however, for various reasons, most operate collectives of a few tens of robots [11,9,12] or at the very most a few hundred robots [13,14]. These collective sizes are primarily limited by robot cost and operational complexity. The cost of an individual robot is perhaps the largest limiting factor for the collective size; for a fixed budget, the lower the robot price, the larger the size of the collective. For example, a popular commercially available robot, the e-puck [12], equipped with an infrared communication ring for collective operations, costs over $1300, and as a result, is usually operated in collectives of about 10 robots. A robot more oriented towards large scale multi-robot research is the Jasmine robot [14]. This robot costs about $130 in parts and has been operated in collectives of 150 robots; however, the robot is not available for purchase.

In addition to cost, the complexity of robot operations, such as turning the robots on/off, charging, controlling, and programming the collective, also plays a role in limiting the size of the collective. For example, a common way to control the power of a robot is to have a switch on each robot to turn the robot on and off [12,11]. If this switch takes three seconds per robot to find and flip, then it would take a single person 50 min to turn on a collective of 1000 robots! Furthermore, if the collective is large enough, the first robot to turn on may actually run out of power before the last robot is turned on. This manual power switch as well as other design decisions can prevent the size of the collective from scaling to large numbers.

To make a robot suitable for large collective sizes, as described in [15], *all the operations of the robot must work on the collective as a whole*, and not require any individual attention to the robot, such as pushing a switch or plugging in a charging cable for each robot. In other words, all collective operations must take a constant time to do, no matter how many robots are in the collective, i.e. the operations are scalable. An example of a scalable operation on a robotic collective is the programming of the I-Swarm robots [16]. In these robots, instead of plugging in a programming cable to each robot in order to update its program, each can receive a program via an infrared communication channel. This allows an overhead infrared transmitter to program all the robots in the collective in a fixed amount of time, independent of the number of robots. Another example of scalable operations is found in [15,14], where instead of manually plugging in each robot to a charger for battery charging, they use an automatic charging dock that allows the robots to charge themselves without human help, thus making the robot charging scalable. An example of a scalable operation regarding power control is found on the Robomote [10], and in sensor networks [17]. Instead of powering off a robot, it is always on, but in a low-power sleep state, ready to turn on if the appropriate command is received. As a result, a Robomote never has to be turned on or off manually, and the entire collective can be turned "on" in a fixed time independent of the number of robots. These sorts of scalable operations are essential for collective operations, but at the same time, they should not dramatically change the robots' capabilities, cost, or ease of manufacturing.

The rest of this paper will introduce a new robot, Kilobot [18], which is a low cost, open-source robot with fully scalable operations. This robot is designed to make testing collective algorithms on hundreds or thousands of robots accessible to robotics researchers. First we describe the hardware design of a Kilobot robot, where its low cost ($14 worth of parts) and quick assembly (5 min) enable large numbers to be produced easily. While these robots are low cost, they still have abilities similar to other collective robots. These abilities include: differential drive locomotion, on-board computation power, neighbor-to-neighbor communication, ambient light sensing, and neighbor-to-neighbor distance sensing. These abilities are achieved at low cost mainly through the use of vibration-based locomotion and a simple range only sensor. Next, we discuss how the operations of a Kilobot robot, such as programming, turning power on and off, battery charging, and starting/stopping programs, do not require any individual attention by a person, and therefore a large collective can be easily overseen by a single operator. Table 1 gives an overview of some of the robot platforms that have been used for testing collective behaviors, and compares them to the Kilobot. While the Kilobot is relatively simple compared to other robots, we believe it capable of running many interesting collective behaviors. This simplicity allows for low cost, which combined with scalable operations allows for collectives much larger than are currently available.

Finally we will demonstrate that despite its simplicity, a Kilobot can still be used for implementing and testing collective behaviors. These demonstrations use a Kilobot test collective of up to 100 robots to implement the popular collective behaviors of foraging, formation control, phototaxis, and synchronization. A video showing these demonstrations is included with this paper (see http://dx.doi.org/10.1016/j.robot.2013.08.006), and available online at www.eecs.harvard.edu/ssr/projects/progSA/kilobot.html. The open-source documentation on how to build and run Kilobots can also be found at that webpage.

2. Kilobot design

Two competing factors were considered when designing the Kilobot robot: the cost and the functionality. The robot needs enough functionality to allow it to perform a wide variety of collective behaviors, while at the same time, it must be simple enough to keep the cost low.

SDASH [3], an algorithm developed to self-assemble and self-heal a collective shape, was chosen as a collective behavior to motivate the Kilobot hardware design. This rather complex behavior requires that the robots have the ability to: (1) move forward, (2) rotate, (3) communicate with nearby neighbors, (4) measure distance to nearby neighbors, and (5) have sufficient memory to run SDASH. We feel that these requirements taken from SDASH also give a good sample of robot capabilities needed for many other collective robot behaviors. Furthermore, to improve Kilobots' ability to operate in large collectives, as well as to make it a more versatile robotic platform, some additional requirements beyond the five from SDASH were added. The additional requirements are that the Kilobot must: measure ambient light levels, display some internal state to assist with debugging, and allow for scalable operations.

While these are not the minimum set of functions needed for a collective robot, they strike a balance between what behaviors a collective of robots is capable of and the cost of that collective. This section describes the design of the Kilobot hardware which has the functionality desired, while also keeping the price low. A general overview of the Kilobot robot is given in Fig. 1. The environment, called the "arena", that these robots are intended to operate in, consists of a smooth, level, reflective table (e.g. a standard dry erase surface) which can be seen in Fig. 2.

2.1. *Locomotion*

One important capability of the Kilobot is that it must be able to move in its environment. The most common locomotion strategy for swarm robots is to use a two-wheeled differential drive, where each wheel is powered by an electric gear motor. While this conventional wheeled locomotion is quite effective, it is relatively expensive. To keep the cost down, the Kilobot uses two sealed coin shaped vibration motors for locomotion. When one of these motors is activated, the centripetal forces generated by the vibrating motor are converted to a forward force on the Kilobot located at the motor's mounting location. The principle of converting the motor

Please cite this article in press as: M. Rubenstein, et al., Kilobot: A low cost robot with scalable operations designed for collective behaviors, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.08.006

Table 1
Comparison of some collective robot systems.

Robot	Cost	Scalable operations	Sensing	Locomotion, speed	Body size (cm)	Battery life (h)
Kilobot	$14[a]	Charge, power, program	Distance, ambient light	Vibration, l cm/s	3.3	3–24
E-puck [12]	$1300	None	Camera, distance, bearing	Wheel, 13 cm/s	7.5	1–10
Jasmine [19]	$130[a]	Charging	Distance, bearing, light color	Wheel, N/A	3	1–2
R-One [5]	$220[a]	None	Visible light, 3d accel, 2d gyro bump, IR sensors, encoders	Wheel, 30 cm/s	10	6
SwarmBot [6]	N/A[b]	Charge, power, program	Range, bearing, camera, bump	Wheel, 50 cm/s	12.7	3

[a] Part cost only.
[b] No price information available.

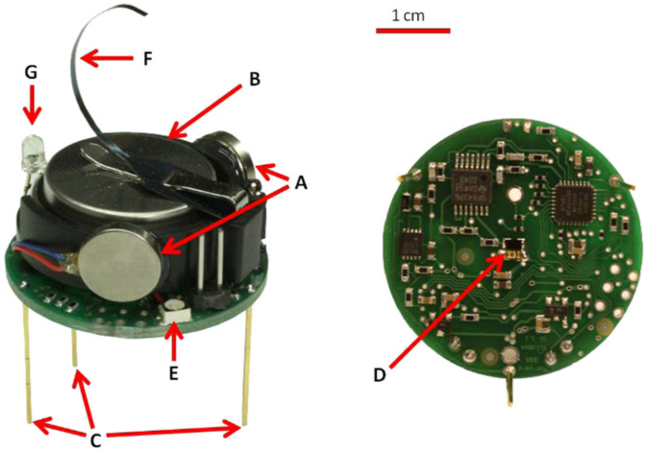


Fig. 1. Isometric (left) and bottom (right) views of a Kilobot. Some key features are: (A) vibration motors, (B) lithium-ion battery, (C) rigid supporting legs, (D) infrared transmitter/receiver, (E) three-color (RGB) LED, (F) charging tab, and (G) ambient light sensor. Note the 1 cm line for scale. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

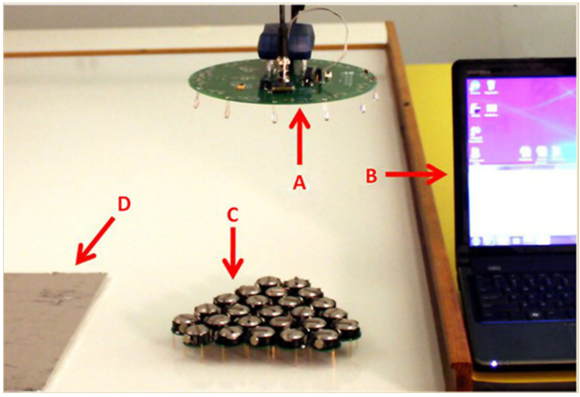


Fig. 2. Picture of the Kilobot arena, including overhead controller (A), control station (B), 25 Kilobots (C), and charging station (D).

vibration to a forward force can be explained using the slip–stick principle, the details of which can be found in [20]. The slip–stick locomotion of a Kilobot was confirmed using high-speed video of the robot's movement. Due to the off-center mounting of the two vibration motors, as shown in Fig. 1, the vibration of one motor alone will cause a rotation of the Kilobot about its vertical axis, while the vibration of the other motor will cause an opposite rotation. By controlling the magnitude of vibration for the two motors independently in a differential drive manner, the robot can move in a continuous range from clockwise rotation, to straight forward, to counter-clockwise rotation. This enables the Kilobot to move approximately 1 cm/s and rotate approximately 45°/s.

One major drawback to using this low cost slip–stick based locomotion, as opposed to wheels with encoders, is that there is no real form of odometry. This makes moving precisely over long distances or for a long time difficult. One way to address this difficulty, which harnesses the power of a collective, is to use the measured distances between neighbors as feedback to correct errors in the robot's movement. As is shown in Section 2.6, this allows the robot to achieve fairly accurate motion control when aided by other robots. Another limitation to this locomotion is that it cannot move over rougher surfaces, requiring a smooth surface such as a dry erase surface to work. While this does limit the environments that the Kilobot can operate in, it dramatically reduces its cost, and still allows for the demonstration of many interesting collective behaviors.

2.2. *Communication and sensing*

A vast majority of collective robot algorithms use robot-to-robot communication and sensing, such as distance and bearing to neighbors, as the main information to drive the behaviors of individual robots. Therefore, it is critical that the Kilobot also be able to communicate with its neighbors and sense some information about its physical relation to its neighbors. In order to keep the robot cost down, the sensing of neighbors only includes distance sensing, not bearing sensing. While bearing sensing is often used with collective robots, for example [19], distance-only sensing is still sufficient for interesting collective behaviors, including SDASH [3].

To communicate with neighboring robots, each Kilobot has an infrared LED transmitter and infrared photodiode receiver, which are located in the center of the PCB and are pointed directly downwards at the table the Kilobot is standing on as shown in Fig. 1. Both the transmitter and receiver have an isotropic emission or reception pattern, which allow the robot to receive messages equally from all directions. Additionally, both the receiver and transmitter are wide-angle, with an angle of half power of 60° from the robot's downward pointing vertical axis. When the transmitter is active, any nearby robot can receive the light emitted by the transmitting robot after it is reflected off the table, as shown in Fig. 3. Messages are transmitted by pulsing the transmitter according to the standard line coding technique. Using this simple communication method, a Kilobot can communicate at rates up to 30 kb/s with robots up to 10 cm (about 6 robot radii) away.

With all robots using the same infrared channel for communication, there is the possibility that two or more robots may try to transmit at the same time. To mitigate this problem a standard carrier sense multiple access with collision avoidance (CSMA/CA) method is used. Even with CSMA/CA, environments with many nearby robots will experience a reduction of the channel bandwidth due to collisions. In an experiment with 25 robots, configured as shown in Fig. 2, the channel could support on average 240 five-byte packets/second, a 32% channel usage.

During any communication between robots, the receiving robot also measures the intensity of the incoming infrared light using two amplifiers, each attached to an analog-to-digital converter built into the microprocessor. These two amplifiers give a low gain amplification of the signal for sensing nearby robots, and

Please cite this article in press as: M. Rubenstein, et al., Kilobot: A low cost robot with scalable operations designed for collective behaviors, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.08.006

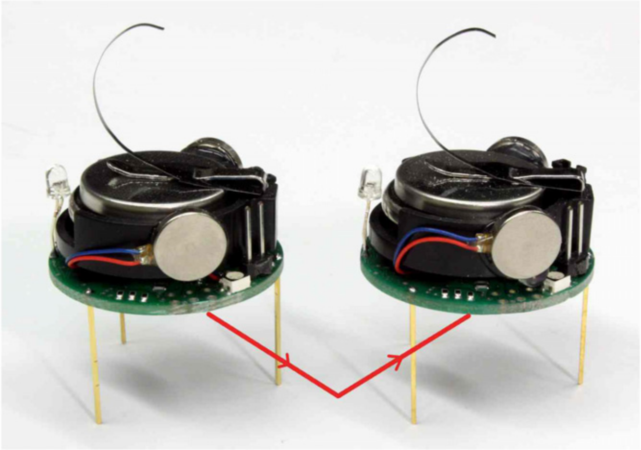


Fig. 3. Illustration showing the reflection path of robot communication.

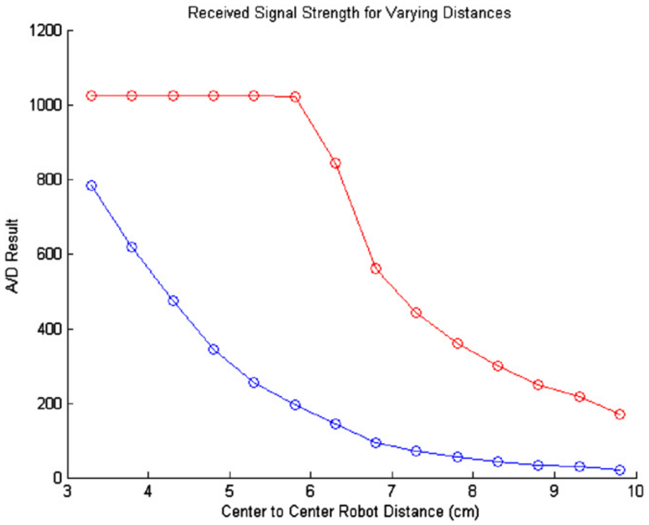


Fig. 4. Received signal strength from incoming messages with varying distance. High gain data shown as red circles; low gain data shown as blue circles. Linear interpolation of data shown as lines. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

a high gain amplification for sensing more distant robots. The incoming light intensity is a monotonically decreasing function of the distance between the transmitter and the receiver; therefore the distance to the transmitter can be calculated by the receiver; see Fig. 4. In practice, the incoming intensity of light is also affected by noise and manufacturing variances, which leads to a sensing accuracy of ±2 mm, and precision under 1 mm.

There is also a visible light sensor on each robot, which can sense the level of ambient light shining on the robot. While this sensor is not used in SDASH, it may be useful for other collective applications such as phototaxis or collective transport.

2.3. Controller

The controller for the robot serves two functions. First, it interfaces with all the low-level electronics such as motors, communication, power circuitry, and the RGB LED (used for displaying information to the operator, seen in Fig. 1). Second, it runs a user-defined robot behavior program. The controller used is an Atmega328 microprocessor, which runs at 8 MHz and has 32 K of memory, sufficient space for running an SDASH controller. Some key features of this controller used in the Kilobot are: two pulse width modulation (PWM) channels used for controlling the speed of the vibrating motors, 10-bit analog-to-digital converters used for measuring the incoming infrared light intensity, self-programmable memory used to update the robot's program (described in Section 3.3), and a low-power sleep mode (see Section 3.1). The program for the robot is written in C, which allows researchers to quickly and easily develop robot behaviors.

Table 2
A summary of Kilobot part cost. Part costs are bulk prices at quantities for 1000 robots.

Category	Cost
Locomotion	$3.12
Power	$3.61
Communication/sensing	$2.20
Control	$2.83
Structure	$1.55
Miscellaneous	$0.74
Total parts	$14.05

2.4. Power system

To power the entire robot, each Kilobot has a 3.4 V 160 mAh lithium-ion battery, shown in Fig. 1. This battery can power the robot for 3–24 h depending on the robot's activity level. Connected to this battery are three voltage regulators and a battery charger. Two of the voltage regulators provide power to the motors and the communication system. Both of these regulators can be switched on and off by the microcontroller, enabling shutdown of the motors and the communication system to conserve power consumption. The third voltage regulator continuously provides power to the microcontroller, and during low-power states (described in Section 3.1) only draws 30 μA. When the battery charger receives 6 Vdc (described in Section 3.2), the charger will begin charging the onboard battery; when the battery is charged, the charging will stop.

2.5. Cost

To allow for large Kilobot collectives, it is critical that each robot be as low cost as possible. The Kilobot design as described uses about $14 worth of parts, which is at least 10 times less than the lowest cost of currently used collective robots [19]. This cost does not include the assembly of components on the PCB, which can be done by a pick-and-place machine. The cost of each Kilobot can be broken down into six categories: locomotion, power, communication/sensing, control, structure (includes PCB and battery holder), and miscellaneous (all other parts, such as the RGB LED). Table 2 gives a summary of the cost of these six categories of Kilobot parts.

The assembly time of a robot can also affect the price of a robot if it is pre-assembled; if not, it can make building a collective difficult and time-consuming. Either way, it is important for the robot to be able to be assembled quickly. To aid in a quick assembly time, most of the robot's components are surface-mount and are placed using a pick-and-place manufacturing robot which costs less than $5. The remaining parts to be assembled are: the battery holder, ambient light sensor, the legs, the motors and the infrared receiver and transmitter. The battery holder and the infrared receiver and transmitter can be assembled by hand. For the remaining parts, custom-made assembly rigs allow for quick and precise alignment and attachment of the motors and the legs. This entire assembly process takes less than five minutes.

2.6. Robot capability demonstrations

To demonstrate Kilobot capabilities, this section shows two demonstrations that provide evidence for the robot's basic functionality: the ability to move within its environment, run a controller, communicate with its neighbors, and measure distance

Please cite this article in press as: M. Rubenstein, et al., Kilobot: A low cost robot with scalable operations designed for collective behaviors, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.08.006

to those neighbors. Additionally, these demonstrations show that when Kilobots sense and communicate with neighbors, they can improve their capabilities beyond what is directly available to them in hardware. In these demonstrations, the improved capabilities shown are position sensing and bearing sensing.

2.6.1. Orbit demonstration

In the first demonstration, a single robot is tasked to travel on a circular path. Lacking odometry, this task is not possible for a single Kilobot, as shown in Fig. 5(C). However, with the assistance of a stationary neighboring Kilobot acting as a marker for the center of the orbit, orbiting is possible, as shown in Fig. 5(A). This stationary robot sends out a message at 1/10th of a second intervals, which is received by the orbiting Kilobot. These messages allow the orbiting Kilobot to compute its current distance to the stationary robot, which is also its distance to the center of its desired orbital path. These distance measurements allow the onboard controller to compute the robot's deviation from the ideal orbit, and using a PD controller, adjust the intensity of both motors' vibrations to correct for it, keeping the robot close to the ideal path.

2.6.2. Path following demonstration

A second demonstration shows a Kilobot following a more complicated path, in this case a "U" shaped path. This path is defined in Cartesian coordinates as a polygon in which the robot is allowed to move. If the robot is inside the polygon, it moves straight, if not, it turns back towards the interior of the polygon. To enable this behavior, three stationary robots are set in the environment in a triangle shape, as shown in Fig. 5(B). These three robots know their position in a coordinate system and communicate that position to the moving robot 10 times a second. The moving robot uses the communicated positions of the three stationary robots, as well as its measured distance to those robots, to trilaterate its own position in the coordinate system. Once the moving robot knows its position in the coordinate system, it can compute if it is located inside the polygon or not. If not, it can also compute which direction to turn in order to move back into the polygon. The position of the moving robot during five attempts at following the path is shown in Fig. 5(B).

3. Scalable robot operations

With a large collective of robots, it can become tedious or even impossible to work with it if the robots require a human operator to interact with each of the robots one at a time. Some of these individual interactions could include pushing the robot's power switch on or off, plugging in a cable to program or charge the robot, and pausing, starting, or stopping the program running on the robot. The following section explains how a Kilobot does not require any individual attention for normal collective operations, i.e., its operations are scalable. To allow a single user to scalably operate the collective, the setup for Kilobot testing and operations has an overhead infrared controller, as shown in Fig. 2. This controller can send infrared messages, using the same methods as in 2.2, to all the Kilobots on the testing arena at once. The overhead controller is in turn controlled by an operator sitting at a computer-based control station.

3.1. Power

To avoid a physical switch to turn the robots "on" or "off", Kilobots use a power control scheme similar to some sensor networks and the Robomote [10]. This power control scheme works by replacing the standard "off" state of the robot, where the battery is disconnected from the robot, with a low-power sleep mode. While in this mode, the battery stays connected to the robot, but the robot powers down most of its electronics and the microprocessor goes into a sleep state for one minute. During this time, the power drawn from the battery by the Kilobot is approximately 100 μW. After one minute, the microprocessor wakes up, turns on the infrared sensor, and for 10 ms it checks if it receives a wake-up message from the overhead controller. To ensure that a message is received during this 10 ms window, the overhead controller transmits the wake-up message every 3 ms for over a minute. If a wake-up message is received by the robot, it switches to the standard "on" mode. If no wake-up message is received, then the robot will go back to the low-power sleep mode, repeating this cycle until a wake-up message is received. If the robot is in the "on" mode, a sleep message sent from the overhead controller will switch it to the low-power sleep mode.

Using this power control scheme, the robot can remain in the low-power sleep mode, waiting to switch to the "on" mode, for more than 3 months on a single battery charge. Furthermore, the entire collective can be switched from the low-power sleep mode to the "on" mode in under one minute, and from the "on" mode to the low-power sleep mode in a few seconds.

3.2. Charging

As described in Section 2.4, when 6 Vdc is applied to the input of a Kilobot's battery charger, it will charge the on board battery until the battery is full. To apply this potential to the charger input, the charging tab is connected to an electrical ground, and the bottom legs are connected to 6 Vdc, as shown in Fig. 6(A). This enables an entire collective to be charged by first placing it onto a conductive surface, which can easily be done by pushing the whole group with a long stick as shown in Fig. 6(B). Next, a conducting board, such as a poster board coated with metallic tape, is placed on top of the collective. Finally 6 Vdc is applied across the bottom conductive surface and the top conducting board, connecting the input of all the chargers to the required voltage. Fig. 6(C) shows a group of Kilobots in this charging configuration which does not require any attention to the individual robots.

3.3. Programming

To change the program running on the Kilobot's microcontroller, the self-programming feature of the Atmega328 is used. This feature allows the main program code, located in the primary sector of memory, to be overwritten by a program written in a separate "bootloader" sector in memory. In this bootloader sector of memory, the Kilobot has a program that receives infrared messages from the overhead controller which contain portions of the new desired main program code. It then writes these portions of code to the appropriate location in the primary sector of memory. The bootloader program has error-checking to ensure that the program placed in the primary sector is complete and error-free. Once the complete replacement program has been received, the bootloader code resets the microprocessor, and the newly loaded main program code begins execution. When the operator desires to put a new program in all of the Kilobots, the operator transmits a "jump to bootloader" message from the overhead controller. When this message is received by the main program code, it moves the program counter to the bootloader section of code, causing the bootloader program to execute. This scalable programming scheme allows all the Kilobots present in the testing arena to load a new program in under one minute. To validate this, a test collective of 100 robots was programmed using this method in 35 s.

3.4. Other control

In addition to programming and power control, the overhead controller is also useful in other aspects of Kilobot operations such

Please cite this article in press as: M. Rubenstein, et al., Kilobot: A low cost robot with scalable operations designed for collective behaviors, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.08.006

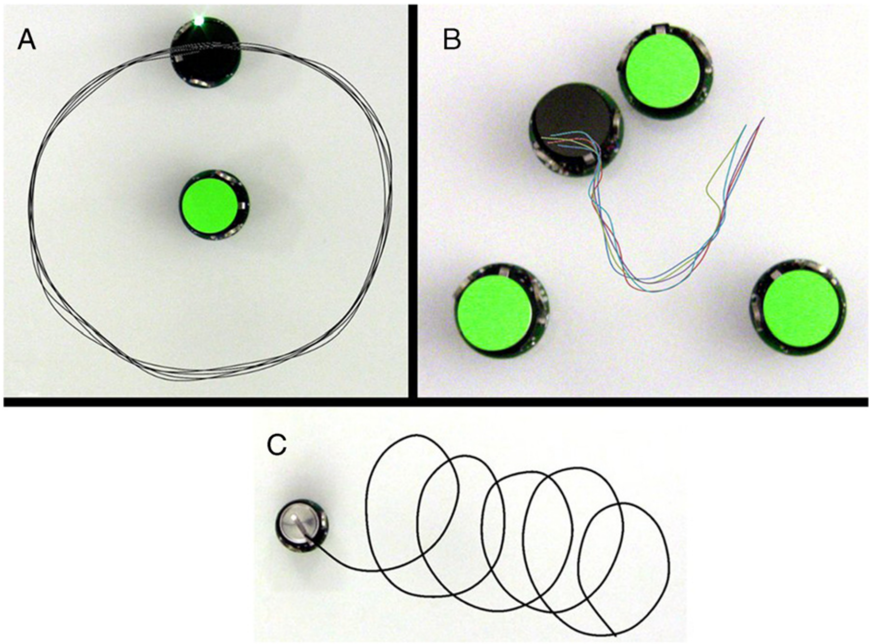


Fig. 5. The path of a Kilobot (black marker) orbiting five times around a second stationary Kilobot (green marker) (A). A "U" shaped path following robot (marked with black) and the three stationary beacon robots (marked with green). The path of the moving robot during 5 attempts at following the desired path is shown (B). The path of a single Kilobot attempting to circle without assistance from other robots (C). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

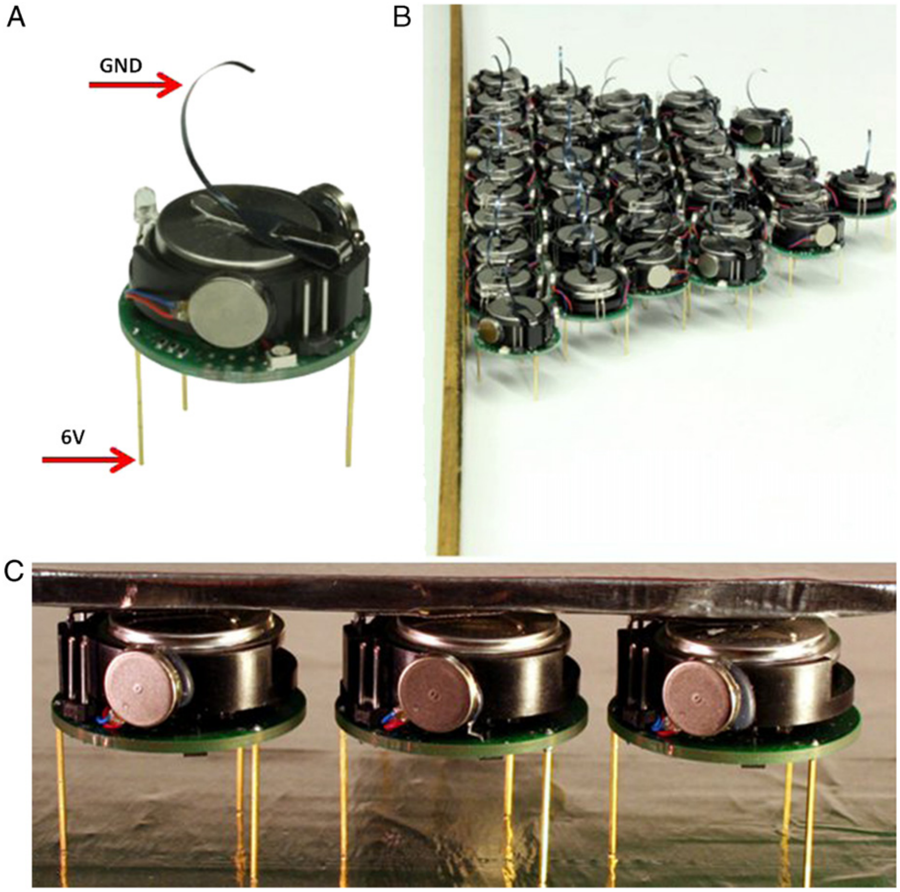


Fig. 6. Connecting to charger input (A); manually pushing robots towards charger (B); a side view of a group of Kilobots charging (C).

Please cite this article in press as: M. Rubenstein, et al., Kilobot: A low cost robot with scalable operations designed for collective behaviors, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.08.006

as querying a robot's battery voltage, and starting, restarting, and pausing the robot's programs. For example, it may be useful for the operator to know the battery voltage of the robots in the collective to determine if they need to be recharged. To do this, the operator sends a message to all the robots via the overhead controller to display voltage. Each robot then displays a color on its RGB LED based on the measured voltage of its battery, displaying green if the battery has more than 90% charge remaining, blue if between 90% and 40% is remaining, and red if less than 40% is remaining. With all robots displaying their charge status, the operator can then look at the collective and determine its overall charge status. In addition, the operator can control the execution of the main program in all Kilobots by issuing commands via the overhead controller to pause, start, stop or restart the main program.

4. Collective behavior demonstrations

While designed to be low cost and simple, the Kilobot is still capable of interesting collective behaviors. To demonstrate this, a test collective of 100 robots was built and used to implement four popular collective behaviors. These behaviors are: ant-inspired foraging, formation control, phototaxis, and synchronization.

4.1. Foraging

Foraging is a popular biologically inspired task in collective robotics, for example [21,22]. The task generally involves robots exploring the environment to find a "food" goal, and delivering it to the "nest". In this demonstration, the foraging algorithm from [23] is implemented directly on a Kilobot collective. Two immobile Kilobots with specialized programs are placed as markers in the environment, as shown in Fig. 7, symbolizing the nest and food locations. These markers constantly transmit a message that tells any robots within the communication range that the marker is the nest or food. For food to be delivered to the nest, a robot must pass close to the food marker, where it "picks up" a unit of food from the food marker, and then the same robot must then pass close to the nest marker, where it "drops off" the food.

In this foraging algorithm, some of the robots can take on the role of a "beacon". The goal of this beacon is to aid the remaining robots, the "walkers", in navigating between the nest and the food. The beacons do this by maintaining two variables, food gradient, and nest gradient. If any beacon sees the food or the nest, it sets its corresponding gradient value to one. Otherwise, the beacon sets its gradient value to be the value of the lowest corresponding gradient it receives from neighboring beacons, plus one. This allows any walker robot to listen to the nearby beacons and determine which beacon is closest to the nest or the food. If the walker does not have food, it moves towards the beacon that is closest to the food. If the walker does have food, it moves towards the beacon that is closest to the nest. A walker can move closer to a desired beacon by sensing its distance to that beacon, and then moving to reduce that distance. While moving, a walker also needs to avoid collisions with other robots. This is done by detecting if any neighbors are less than 1 cm away, and if so, turning away until it no longer is. This overall movement strategy of moving towards the desired beacon while avoiding collisions allows the walkers to travel from the nest to the food and back. If the food has not been reached yet and the walker senses a low density of nearby beacons, then it will become a beacon, expanding the fronter of beacons in the environment. Once beacons, the robots will never switch back to a walker.

Initially, as shown in Fig. 7(A), all the robots are placed next to the nest marker and have no information about the location of the food marker. They all start in the walker role and spread out from the nest. As they spread out and explore the environment, some walkers choose to become beacons. These beacons are created to provide a connection between the exploring walkers and the nest. If, as shown in Fig. 7(B), one or more beacons see the food, then the remaining walkers will be capable of navigating to the food and nest, as shown in Fig. 7(C). For a more detailed explanation of this algorithm, please see [23].

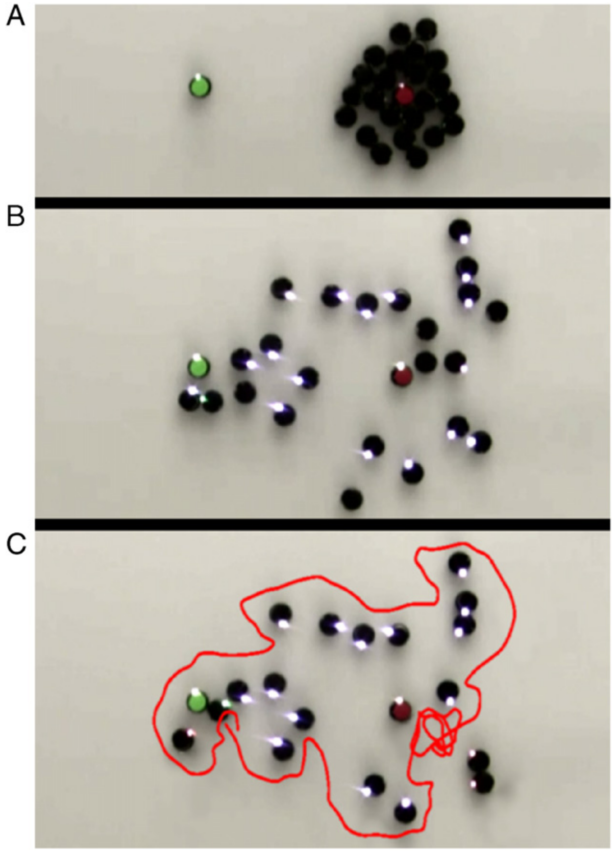


Fig. 7. Initial start of foraging demo: note food (green marker) and nest (red marker) (A). Ant algorithm after finding food: note beacons (RGB LED on) and walkers (RGB LED off) (B). The path of a single walker from the food, to the nest, and back to the food (C). Note: the other walkers in (C) have been digitally removed for clarity. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.2. Formation control

Formation control, another biologically inspired behavior, is often applied to robotic collectives, particularly in the control of unmanned vehicles. This demonstration shows "follow-the-leader", a popular formation control behavior often applied to robot collectives [24], and implements it on 6 Kilobots. In this demonstration, the Kilobots start lined up facing the leader. The leader then moves in its forward direction, and all the followers follow along approximately the same path. Note that as seen in Fig. 8, without positional information, the leader does not always move precisely forward; however, the other Kilobots still follow the leader.

To initialize the Kilobot collective for following the leader, each robot is assigned an ID, where the ID of the leader robot is 1. The ID of the remaining robots is one greater than the robot immediately in front of it. This allows the robots to know the ID of the robot immediately in front and behind it. Additionally, the front robot knows that it is the leader (its ID is 1), and the last robot, the tail, knows that it has no robots behind it (it sees no ID greater than its own). All robots can operate in two states, move and wait. Initially, if the robot has an even ID, it starts in the wait state, and if its ID is odd, it starts in the move state. During the entire demonstration,

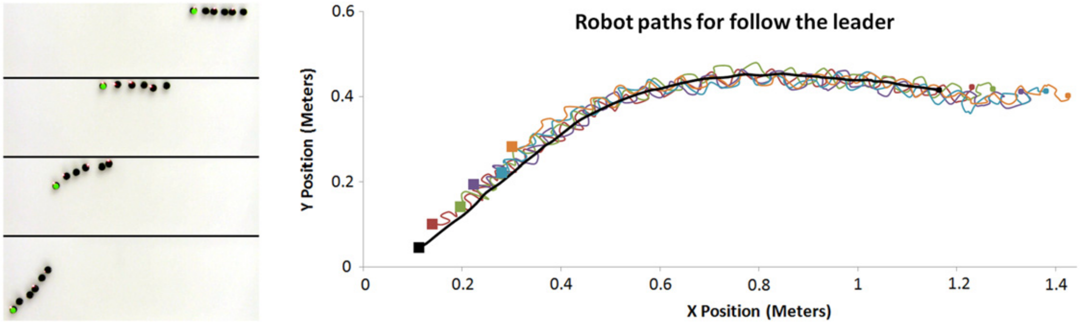


Fig. 8. Screenshots of Kilobots following the leader is shown on the left. The starting frame is on top, ending frame is on bottom. The leader is marked with green. On the right, the paths of five Kilobots following the leader (black line is the leader's path) are shown. Starting locations are marked as circles, ending locations marked as squares. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

each robot will constantly communicate this state and its ID to its neighbors. A robot in the wait state will stay stationary and wait until the robot immediately in front of and behind it are also in the wait state. When this condition is met, the robot will then switch to the follow state. The leader and tail robots only wait for the robot in back or in front of it respectively. During the follow state, each robot will measure the distance to the robot directly in front of it, and move to reduce this distance. Once this distance is less than one cm, the robot will stop moving and switch to the wait state. Since the leader has no robots in front of it, when it is in the follow state, it moves forward until it is four cm away from the robot directly behind it, then switches to the wait state. There are some additional nuanced rules for switching states to prevent premature state changes, but for the sake of brevity, these will not be discussed here. These two behaviors allow robots to move towards the leader robot, while at the same time not move so fast as to break communications with the robot directly behind it. The result is that the collective of Kilobots follow the leader, as shown in Fig. 8.

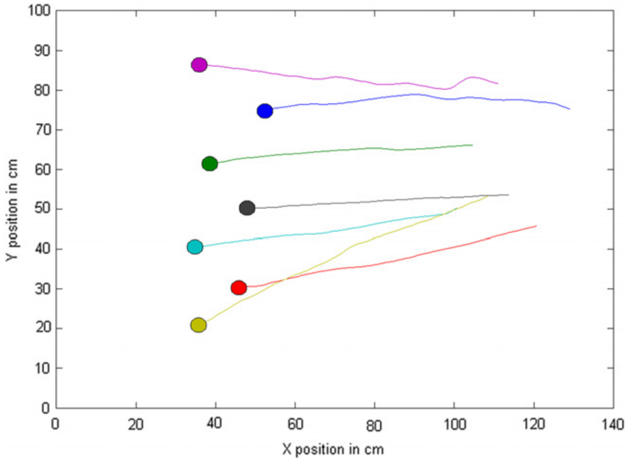


Fig. 9. The tracks of 7 robots starting at the circles on the left, moving towards a light source on the right.

4.3. *Phototaxis*

The ability to move towards light, called phototaxis, is often used in robotics because it allows a robot to accomplish a global task, such as moving to a desired location [25]. This behavior only requires a simple sensor to detect the light, and therefore can be implemented on even the simplest of robots [26]. To accomplish phototaxis, a Kilobot uses its ambient light sensor to determine what the light intensity is at its current location, and using movement it tries to increase its sensed light intensity. This is done by commanding its motors to move it in a broadly sweeping clockwise or counter-clockwise turn until the light intensity value is no longer increasing. Once the light value stops increasing, it switches the direction of its motion from clockwise to counter-clockwise, or from counter-clockwise to clockwise. As the robot monitors its light sensor and switches directions whenever light intensity stops increasing, it will move towards the light source. Fig. 9 shows the tracks of seven Kilobots making progress towards the light source using this behavior. Fig. 10 shows this behavior implemented on 100 Kilobots, moving them towards the goal of the light source.

4.4. *Synchronization*

Synchronization is often used when coordinating simultaneous actions between many entities, such as robots or sensor networks. A popular method for synchronization [27] is based on how fireflies synchronize their periodic light emissions in nature. In this work, each entity maintains a periodic clock that when it reaches its maximum value, the clock resets, and the entity sends a signal to its neighbors. Whenever a neighbor receives a signal, it adjusts its local clock according to when it was received. This adjustment is enough to eventually synchronize all the entities. A modification of this approach [28] is intended to operate on systems that use a communication channel where contention and delays are possible, which is true for the Kilobot system. This demonstration uses an approach closely related to [28] to synchronize the blinking of the RGB LED on 100 Kilobots. Fig. 11 shows the current number of robots blinking in the whole collective vs time and demonstrates that the initially unsynchronized collective becomes synchronized.

5. Lessons learned

After working with the Kilobot system, we learned some lessons about designing and operating such a system that could aid future large scale robot systems. One lesson learned is that while calibration of sensors and motors is included in assembly time, the precision required makes calibration more tedious and harder to automate. Future large scale robot collectives should require no manual calibration, by either calibrating automatically, or by relying on control that does not require any calibration whatsoever.

While the operations of Kilobot are scalable, a less obvious lesson learned deals with scaling the handling the robots. For example Kilobots have exposed PCBs and batteries, so if they were just placed in a large pile there would be risk of electrical shorts. Additionally, Kilobots must be standing correctly; if they are laying on their side, then they will not function properly. This means that to move the robots from one table to another, each robot must be carefully picked up by hand and moved. This causes the simple task of moving the Kilobots from one table to another to take many hours. It would be advantageous for future swarm robots to not require this careful handling.

Please cite this article in press as: M. Rubenstein, et al., Kilobot: A low cost robot with scalable operations designed for collective behaviors, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.08.006

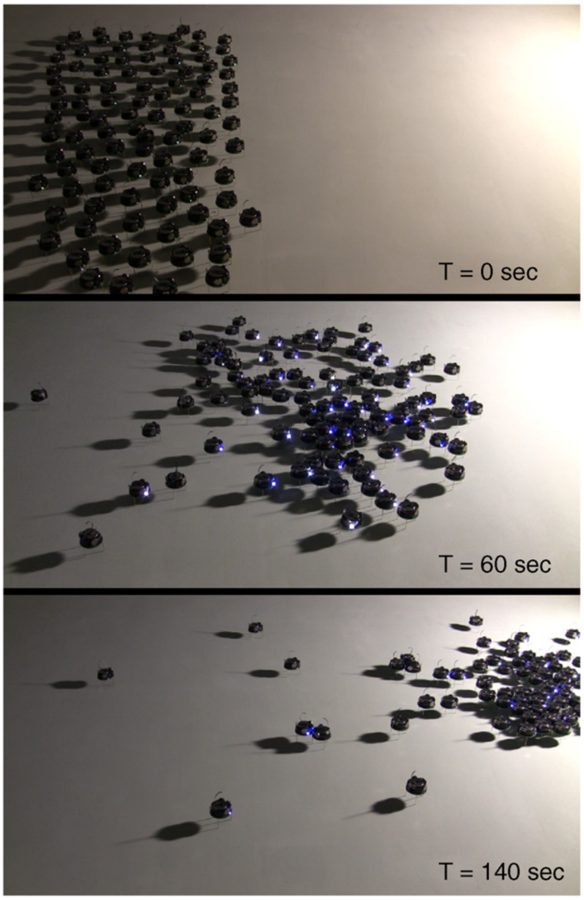


Fig. 10. Three screen captures of 100 robots starting on the left of the screen and moving towards a light source on the right.

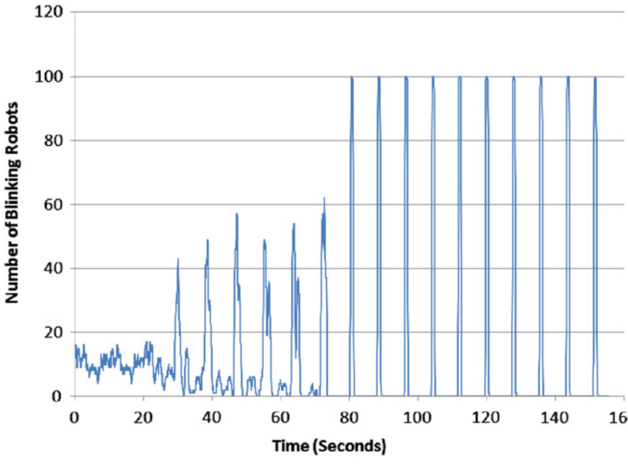


Fig. 11. The number of robots blinking its LED vs time. Initially the robots blink at random times, but over a period of 80 s they become synchronized.

6. Conclusion

In this paper, we have presented the Kilobot robot, a $14 open-source robot designed specifically for operation in a large collective. While the Kilobot is a relatively simple and low cost robot, we have shown it to be a viable collective robotic system by demonstrating four popular collective multi-robot behaviors using a 100 Kilobot test collective. The robot design and collective operations are carefully considered in order to make it easy to manufacture and operate a large collective of Kilobots. To keep the robot cost low, its design uses an unusual locomotion technique (vibration-based) and relatively simple sensing (distance but no bearing), both of which dramatically reduce the robot cost without sacrificing its functionality. In the future, we plan on building a 1024 Kilobot collective and using it to design and test additional collective behaviors such as shape self-assembly and collective transport.

References

[1] D. Arbuckle, A. Requicha, Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: algorithms and simulations, Autonomous Robots 28 (2) (2010) 197–211.

[2] K. Støy, R. Nagpal, Self-repair through scale independent self-reconfiguration, in: International Conference on Intelligent Robots and Systems, 2004.

[3] M. Rubenstein, W. Shen, Automatic scalable size selection for the shape of a distributed robotic collective, in: International Conference on Intelligent Robots and Systems, 2010.

[4] K. Galloway, R. Jois, M. Yim, Factory floor: a robotically re-configurable construction platform, in: International Conference on Robotics and Automation, 2010.

[5] J. Everist, K. Mogharei, H. Suri, N. Ranasinghe, B. Khoshnevis, P. Will, W. Shen, A system for in-space assembly, in: International Conference on Intelligent Robots and Systems, 2004.

[6] A. Howard, L. Parker, G. Sukhatme, Experiments with large heterogeneous mobile robot team: exploration, mapping, deployment and detection, International Journal of Robotics Research 25 (5) (2006) 431–447.

[7] W. Burgard, M. Moors, D. Fox, R. Simmons, S. Thrun, Collaborative multi-robot exploration, in: International Conference on Robotics and Automation, 2000.

[8] M. DeRosa, S. Goldstein, P. Lee, J. Campbell, P. Pillai, Scalable shape sculpting via hole motion: motion planning in lattice-constrained modular robots, in: International Conference on Robotics and Automation, 2006.

[9] K. Gilpin, A. Knaian, D. Rus, Robot pebbles: one centimeter modules for programmable matter through self-disassembly, in: International Conference on Robotics and Automation, 2010.

[10] G. Sibley, M. Rahimi, G. Sukhatme, Robomote: a tiny mobile robot platform for large-scale sensor networks, in: International Conference on Robotics and Automation, 2002.

[11] M. Jorgensen, E. Ostergaard, H. Lund, Modular atron: modules for a self-reconfigurable robot, in: International Conference on Intelligent Robots and Systems, 2004.

[12] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. Zufferey, D. Floreano, A. Martinoli, The E-Puck, a robot designed for education in engineering, in: The Conference on Autonomous Robot Systems and Competitions, 2009.

[13] G. Caprari, P. Balmer, R. Piguet, R. Siegwart, The autonomous micro robot alice: a platform for scientific and commercial applications, in: Proceedings of the Ninth International Symposium on Micromechatronics and Human Science, 1998, pp. 231–235.

[14] S. Kernbach, R. Thenius, O. Kernbach, T. Schmickl, Reembodiment of honeybee aggregation behavior in artificial microrobotic system, Adaptive Behavior 17 (3) (2009) 237–259.

[15] J. McLurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau, B. Schmidt, Speaking swarmish: human–robot interface design for large swarms of autonomous mobile robots, in: Association for the Advancement of Artificial Intelligence Spring Symposium, March 2006.

[16] R. Casanova, A. Dieguez, A. Sanuy, A. Arbat, O. Alonso, J. Canals, M. Puig, J. Samitier, Enabling swarm behavior in mm^3 sized robots with specific designed integrated electronics, in: International Conference on Intelligent Robots and Systems, 2007.

[17] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, System architecture directions for networked sensors, in: Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems, 2000, pp. 93–104.

[18] M. Rubenstein, C. Ahler, R. Nagpal, Kilobot: a low cost scalable robot system for collective behaviors, in: International Conference on Robotics and Automation, 2012.

[19] J. McLurkin, A. Lynch, S. Rixner, T. Barr, A. Chou, K. Foster, S. Bilstein, A low-cost multi-robot system for research, teaching, and outreach, in: Distributed Autonomous Robotics Systems, 2010.

[20] P. Vartholomeos, E. Papadopoulos, Analysis, design and control of a planar micro-robot driven by two centripetal-force actuators, in: International Conference on Robotics and Automation, 2006.

[21] R. Vaughan, K. Støy, G. Sukhatme, M. Matarić, Whistling in the dark: cooperative trail following in uncertain localization space, in: International Conference on Autonomous Agents and Multiagent Systems, 2000.

[22] K. O'Hara, V. Bigio, E. Dodson, A. Irani, D. Walker, T. Balch, Physical path planning using the GNATs, in: International Conference on Robotics and Automation, 2005.

[23] N. Hoff, A. Sagoff, R. Wood, R. Nagpal, Two foraging algorithms for robot swarms using only local communication, in: IEEE International Conference on Robotics and Biomimetics, 2010.

[24] M. Gupta, J. Das, M. Vieira, H. Heidarsson, H. Vathsangam, G. Sukhatme, Collective transport of robots: emergent flocking from minimalist multi-robot leader-following, in: International Conference on Intelligent Robots and Systems 2009.

Please cite this article in press as: M. Rubenstein, et al., Kilobot: A low cost robot with scalable operations designed for collective behaviors, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.08.006

[25] A. Christensen, M. Dorigo, Evolving an integrated phototaxis and hole-avoidance behavior for a swarm-bot, in: Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems, MIT Press, Cambridge, MA, 2006, pp. 248–254.

[26] K. Shirai, Y. Matsumoto, S. Koizumi, H. Ishiguro, 1 DOF swimming robot inspired by bacterial motion mechanism, in: International Conference on Robotics and Biomimetics, 2008.

[27] R. Mirollo, S. Strogatz, Synchronization of pulse-coupled biological oscillators, SIAM Journal on Applied Mathematics 50 (1990) 1645–1662.

[28] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, R. Nagpal, Firefly-inspired sensor network synchronicity with realistic radio effects, in: SenSys 2005.

Michael Rubenstein is currently a postdoctoral fellow working on collective robotics and control in the Self-Organizing Systems Research Group at Harvard University. He received his Ph.D. from the University of Southern California. His thesis topic was on self-assembling and self-healing algorithms for robotic collectives.

Christian Ahler works as an Electro-Technical Engineer with the Wyss Institute at Harvard University in Boston. With a B.Sc. in electro-mechanics from USD and an M.Sc. in analog and wireless electronics from DTU, Christian now focuses on developing and maturing bio-inspired robotic platforms for the Self-Organizing Systems Research Lab at Harvard. Among others, he is involved in the hardware design of the Kilobots, ultrasound ranging systems for the Termes project, interface electronics for novel orthotic sensors, and is teaching an embedded hard and software class for graduate students.

Nick Hoff is a scientist at Raytheon BBN Technologies, Distributed Systems research group. Before coming to BBN, Dr. Hoff earned his Ph.D. studying leaderless algorithms for swarms of hundreds of small robots. His approach to swarm algorithms includes distributed decision making and measurement, not only physical control. Before his Ph.D., Dr. Hoff studied Aerospace Engineering, working on the SPHERES project. His research focused on small multi-satellite systems, specifically autonomous docking algorithms. Dr. Hoff was responsible for the full engineering stack hardware design and fabrication, electrical design and integration, and algorithm implementation and testing.

Adrian Cabrera received his B.Sc. in Microengineering in 2010 from EPFL, where he is currently pursuing an M.Sc. also in Microengineering. He is currently a visiting scholar at the Wyss Institute in Harvard University where he is conducting research in collective transport using large swarms of robots. His research interests include collective robotics and autonomous vehicles.

Radhika Nagpal is a professor in Computer Science at the Harvard School of Engineering and Applied Sciences. She is also a core faculty member of the Harvard Wyss Institute for Biologically Inspired Engineering, where she co-leads the BioRobotics Platform. She was a graduate student and postdoc lecturer at the MIT Computer Science and Artificial Intelligence Laboratory and a member of the Amorphous Computing Group and the Bell Labs GRPW graduate fellowship program.

Please cite this article in press as: M. Rubenstein, et al., Kilobot: A low cost robot with scalable operations designed for collective behaviors, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.08.006