Proceedings of the 2014 IEEE
International Conference on Robotics and Biomimetics
December 5-10, 2014, Bali, Indonesia

A Swarm Framework for Teaching Elementary Addition Operations.

Zhengwei Hui, Juan Rojas, Li Lin, HoLeung
Ting, ChenYu Zhao.
Sun Yat Sen University. School of Software
Guangzhou, Guangdong, 510006, China

Abstract— **The advent of low-cost, functional robots has promoted swarm robotics research. However, an area of research that is yet untouched is how these types of robots could aid in teaching STEM subjects. In particular, there is little work concerning mathematics at the elementary level. The Kilobot robot's size, cost, and functionality offers a good test-bed to explore its use as a pedagogical tool. This work presents a swarm framework to teach elementary mathematical addition operations to children. A state-based synchronous algorithm was used to enable robots to represent both operands and result digits in mathematical operations, all the while exploiting the robots sounds, motion, and illumination to enhance children's learning factors. The system is demonstrated in the V-REP simulation environment to demonstrate the feasibility of the approach.**

I. INTRODUCTION

The advent of low-cost, swarm-like, decentralized but closely cooperating multi-robot systems has started to grow in the last decade [1], [2]. Recently, robots like the Kilobots ("KBs") [3] and the Droplets [1], have enabled researchers access useful functionality at low-costs, leading to an increase in size of swarms from tens to hundreds [4]. Swarm robotics is characterized by simple controllers working together in a scalable and decentralized manner that exploits local robot interactions [5]. Swarm robots have been used to explore collective behaviors as diverse as: collective transport [6], [7], collective construction [8], exploration [9], path formations [10], [11], and more.

In terms of robotics in education, there have been a wide variety of efforts to use robots as an educational tool. Educational robotics was founded on the learning principles of constructionism proposed by Papert [12]. In [13], constructivist principles were used to engage children to learn programming through interaction by using the LEGO Technic kit. Since this time, a great number of robotic initiatives have been attempted, but only a few, like the FIRST and RoboCup competitions have been widely successful. Some research has focused on the development of appropriate curricula to teach robotics at collegiate levels [14], [15] and long-term progressive learning framework's [16]. Others have examined robotics at the kindergarten level, where a social robots is used to teach spatial relations like geometrical thinking [17] or language, logic, and creativity, etc. [18]. However, there seem to be no examples of using robotics to teach elementary mathematical operations.

Teaching elementary mathematical operations could be done indirectly through a social robot, however, we are interested in using the robot's motions and interactive capabilities to be the agents of the operations themselves. Having an adequate robot test-bed is not the only challenge, there are a wide range of factors that render a robotics-based elementary mathematical education kit very challenging. One must first have a pedagogical and appropriately designed mathematical curriculum; the identification of learning factors that can be measured and evaluated to determine if the teaching framework is useful and effective, a robot framework and system that is fit to implement the desired functionality flexibly and in a scalable manner. These elements are imperative, such that, as stated in [19], the intrinsic motivation brought about by the technology will wear off, or even worse, that the approach will in fact hurt the learning process of the students and detract students from the subject.

This work focuses on the implementation of a swarm-like robot system to effect elementary math operations such as addition in a flexible and scalable and flexible way. Due to the scope of this work, details on curriculum development, learning factors, and evaluation will not be considered in this paper.

To this end, the KB robot was selected as the robot of choice due to its cost, size, and functionality. Furthermore, a physical environment for mathematical operations was designed and named the Teaching Board. A state-based synchronous algorithm is used to enable the robots to move according to a given problem and change from operands to result through motion in math operations. This enables the children to count the robots as digits, all the while the robots seek to engage children through its motion, sound, light, and board placement to enhance a child's learning factors. The system is demonstrated in the V-REP simulation environment [20] to demonstrate the feasibility of the approach.

The paper is organized as follows: in Sec. II, the Kb robot is introduced, in Sec. III the System Design, namely the Teaching Board is introduced; in Sec. IV, the driving algorithm is presented; in Sec. V, the simulation environment and experimentation is described, and finally in Sec. VI the Conclusion is presented.

II. THE KILOBOT ROBOT

The Kb is a small (3.3cm diameter) robot designed for swarm-like collective behaviors. The system is capable of scalable operations, mobility, neighbor-to-neighbor communication and distance computations, and has sufficient memory to execute algorithms [6]. In particular, the KB's scalable operations allow it to power up, charge, or program hundreds or even thousands of robots in under one minute. The robot uses differential drive locomotion through two vibration motors to move in 2 DoF: linearly forward at speeds of 1cm/s as well as rotate at angular speeds of $\pi/4$ rad/s. Robots communicate through an LED-emitter and IR receiver at 30kb/s and upto 10cm away. Robots perform distance computations by measuring the incoming light intensity.

III. System Design

A. The Teaching Board

An arena known as the Teaching Board (TB) was devised to house all aspects of a mathematical operation for single digit calculations (one's digit) including: operands, operator, result, and a carry number box. A single digit set-up is shown in the V-REP simulation environment in Fig. 1. Single digit operations can at time become multi-digit operations and employ a carry digit in such cases. For instance, $3*4=12$ uses two single digits and yields a double digit result. While our system is designed to consider multi-digit operations as a result of a carry number, this paper will not consider carry operations. Likewise, we will not consider fractions or negative numbers. The TB consists of counting sections.

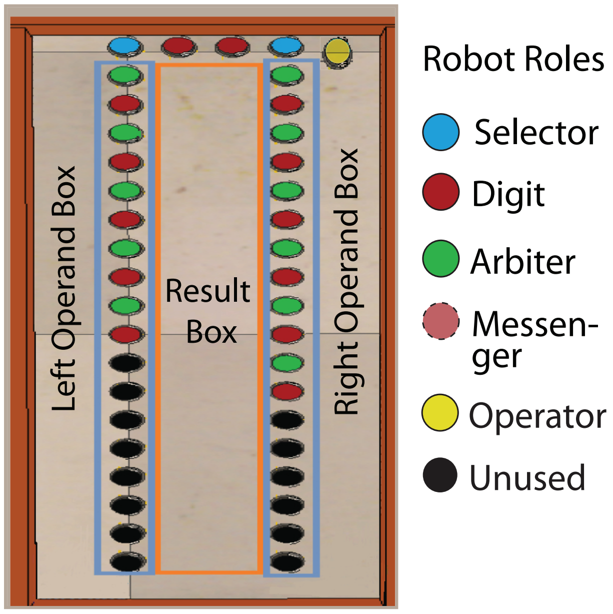

Fig. 1. The teaching board for a single digit operation contains a right and a left operand box at each edge. Each operand box contains 20 Kilobots. The center area also contains a box for the operation's result. Also, four robot roles are shown: Selectors appear in blue, Digits appear in green, Arbiters appear in red (in-between Digits) and Messengers also in red, can be located between Selectors or Digits.

Each counting section is designed for single digit operations and is composed of a result box and two separate counting boxes–a left and a right operand box. At the outer border of each operand box twenty KBs sit at their home position (the number of KBs will be explained later as it relates to the synchronous communication strategy that we use). Given that the KBs have a short communication range (10cm not including it's radius) and that they can only communicate with their immediate neighbors, the distance between adjacent robots is kept to 3.3cm. KBs here represent the operand digits that form part of an operation. The result box occupies the counting section's center area where the operation's result is represented by a participating number of KB digits. We also consider double digit operations without a carry. Such operations can be the result of: (i) two single digit addition or multiplication that lead to double digits, like: $6+5=11$ or $3*4=12$, or (ii) double digit operations that lead to double digit results. For our current system, without a carry, we can maximally represent a maximum value of 18. For carry situations, a second counting section is needed. In general, n counting sections, require $n-digit$ additions or multiplications and an additional counting section for a carry or borrow bit. New counting sections are linked through KB messengers to maintain communication and synchronized motions in the system (KB roles will be discussed in Sec. III-B) across counting sections. The operator was chosen to be represented by a paper note that contains any of the four operator symbols: $\{+,-,x,/\}$.

Given an operation, the KB controller will execute the calculation and drive the appropriate number of KBs from the operand boxes to the Results Box. Once that operation is terminated the KBs return to their home position.

B. Robot Roles

Our system assigns five different roles to the KB robots: Selectors, Digits, Arbiters, and Messengers.

1) Selectors: There are two Selector robots per counting section. Selectors stand atop of the left and right counting boxes respectively. They are designed to select the appropriate number of Digit KB robots according to a given operation. For example, in a $1+1$ operation, the left Selector chooses one KB Digit robot to represent the left operand, the same is done for the right Selector robot.
2) Digits: They exist in both the left and right counting boxes. They play the role of operand digits in the TB. A certain number of them, as assigned by the commander, move towards the Results Box when an operation is executed. Digit KBs will light up sequentially and beep to indicate that the result execution of an operation is complete and the student can count the number of digits in the Results Box. Digits can also have three types of attributes: (i) they can be Leaders (of an Operand Box queue), (ii) they can belong to the Left or the Right Operand groups, and (iii) they can be associated with Arbiter's of type A or type B.
3) Messengers: They are message forwarding KBs. Given that the KB communication range is short, Messengers pass Selector generated messages through neighboring KBs in the home position.
4) Arbiters: These KBs send distance messages to the Digits KBs and assist them in maintaining an orbital path around the Arbiters. This is a technique we use to move selected KBs in a synchronized motion. Arbiters

can be classified as type A or type B KBs. This classification is useful to pair them up with specific Digit KBs during navigation operations.

5) Operators: These robots declare the type of arithmetic operation for the Digit KBs.

Note that Operand Boxes consist of a 19 KB queue: 10 Arbiters and 9 Digits between a pair of Arbiter KBs. The Results Box contains the two selected operand digits and presents the calculation result. Also 2 Messenger KBs are placed between the Selectors to bridge communication between both columns.

C. Robot States

Each of the robot roles presented in the previous sections has a specific set of states. Each state is constrained to a single input and a single output. This constraint yields to a sequential behavior in the robots, leading to simple behaviors and state transitions even when the number of robots is increased.

The Selector KBs have 3 states: {WAIT, GATHER and GO}. In the WAIT state, Selectors await for messages from the controller. Upon receiving a message its state transitions to GATHER, at which point the Selectors emit a count message to Digit KBs that determines the appropriate number of Digit KBs for the operation in the Operand boxes. This count message returns to the Selectors. Upon receipt of the count message, the Selector State transitions to the GO State and instructs Digit KBs to execute the operation.

The Digit KBs have nine states: {WAIT, GATHER, GO, STOP, COUNT, CARRY, WAIT_OTHER_GROUP, RETREAT, STOP}. During WAIT, Digit KBs wait for a GATHER signal to transition to the GATHER state and select the appropriate number of Digits. At this point, they wait for the GO signal at which the Digit KBs move to the Results Box and stop at an exact location therein. The sequence of instructions between the Left Operand Box and the Right Operand Box is not simultaneous but rather sequential. The left box initially receives the signal and the latter is then transmitted to the right. When all Digit KBs stop at a pre-assigned positions, the COUNT state starts, leading Digit KBs to turn on their LEDs and beep from top-to-bottom and from left-to-right. If a counting section generates a carry bit, the Digit KBs transition to a CARRY state and would use Messenger KBs to pass the Carry signal to the next Counting Section. After finishing COUNT (-CARRY) state, the Digit KBs on the left transition to WAIT_OTHER_GROUP (WOG) and wait until the right side is ready to also commence the RETREAT (RET) motion that will drive the KBs to their home position. Upon reaching the home position, the state transitions to the STOP state. At this point, the whole process repeats again for a new commanded operation.

The Arbiter KBs have two states: {FORWARD, NAVIGATE}. The FORWARD (FWD) state renders Arbiters into Messenger KBs and simply forward signals. For example, when the Digit KBs are gathering, the Selector KBs will send a count ID to the Digit KBs. If the ID=7, seven Digit KBs will be selected. Each time a Digit KB is pinged, the count variable decrease until the ID value is zero. The NAVIGATE (NAV) state, on the other hand, has Arbiters emit distance signals to their immediate Digit KB neighbors to aid in navigation.

The Messenger KBs only have the FORWARD state. They serve the same role as the same state for the Arbiter KB but in this case it is used for when a carry bit is generated. In such instances, the Messenger KBs deliver the carry signal to the next counting section.

The operator KB has one state, the OPERATOR state. It is used to determine the arithmetic operation for the Digit KBs.

As a final note, it is important to clarify, that in order to meet our one-input, one-output constraint, some states are divided into several sub-states. For example, the GATHER state of Digit KBs has three sub-states:{GATHER1,GATHER2,GATHER3}. The list of sub-states can be found in Table II-VI.

D. System Progression

In this section, we provide a high-level description of how the entire system progresses. The system development can be tracked by primarily following the Digit KBs state development. These KBs play the primary role in the system. Their behavior can in turn be abstracted into three general stages: Prepare, Ready, and Execute. Preparation involves actions that are being taken by individual KBs. Ready indicates that all KBs have finished their preparation state and are ready for execution. And the last stage triggers the required action. Table I visualizes the sequential progress of the system which is bootstrapped by the Digit KBs.

TABLE I
SYSTEM PROGRESSION.

Stage	Prepare	Ready	Execute
Gathering	Get ID	All KBs ready	Move
Moving	Stop in Results Box	All KBs stopped	Count
Counting	KBs Flash	All KBs flashed	Carry
Carrying	Msg new Count Sec.	Digit KB added	Retreat
Retreating	KBs Retreat	All have retreated	Wait

E. Robot Signals

KB messages consist of three bytes. We encode these messages such that the first byte carries data, the second represents the robot state, and the third represents the sending robot's role. KBs can only send and receive only one signal per state. Table II, gives an example of such representations. Additionally, the state sequence for each to the KB roles can be seen in Tables II-VI.

IV. Algorithm Implementation

The system implements a stated-based behavior control algorithm. Each role has its own state set. KBs change its state according to a received message. For each state, there is only one acceptable input. Each KB will wait for a

TABLE II

ROBOT SIGNAL REPRESENTATIONS.

Process	Code	Example	Note
WAIT	0	(0,0,3)	Digit Waiting.
GATHER	1	(0,1,4)	Selector Gathering
MOVE	2	(0,2,3)	Digit Going
COUNT	3	(0,3,3)	Digit Stopped
CARRY	4	(0,4,3)	Digit Counting
RETREAT	5	(0,5,3)	Digit Retreating

TABLE III

SELECTOR STATES: (2ND BYTE)

STATE	CODE	INPUT	OUTPUT
WAIT	0	(0,0,3): GATHER	(0,0,4)
GATHER	1	-	(X,1,4): X operand
GO	2	Abandon	

TABLE IV

ARBITER STATES: (2ND BYTE)

STATE	CODE	INPUT	OUTPUT
FWD1	11	(X,1,Y)	(X,1,Y)
NAV1	21	(X,3,Y): To FWD2	(0,2,A/B)
FWD2	12	(X,3/4,Y): If 2nd byte=4 to FWD3	(0,3,A/B)
FWD3	13	(X,4/5,Y): If 2nd byte=4 to NAV2	(X,4,A/B)
NAV2	22	-	(0,5,A/B)

TABLE V

MESSENGER STATES: (2ND BYTE)

STATE	CODE	INPUT	OUTPUT
FWD1	11	(X,1,Y)	(X,1,Y)
NAV	21	(X,3,Y): To FWD2	(0,2,A/B)
FWD2	12	(X,3/4,Y): If 2nd byte=4 to FWD3	(0,3,A/B)
FWD3	13	(X,4/5,Y): If 2nd byte=4 to NAV2	(X,4,A/B)
NAV2	22	(X,0,3/4): To FWD1	(0,5,A/B)

TABLE VI

DIGIT STATES: (2ND BYTE)

State	Code	Input	Output	Leader Behavior
WAIT	0	(0,0,4):To GATHER1 (X,Y,Z). If X>0: To GATHER1	(0,0,3)	-
GATHER1	11	(X,1,Y): Id = X, to GATHER2	(0,0,4): Others to GATHER1	Y=4, isLeader=Yes
GATHER2	12	(0,1,X): To GATHER3	(id-1,1,3)	-
GATHER3	13	(0,2,3): To Go	(0,1,3)	if(isLeader==T): to Go
GO	2	Move & stop at given location: To STOP	(0,2,3)	-
STOP	3	(X,3,Y): If X=0, to COUNT1.	If X >0: (id-1,3,3)	if(isLeader==T): Output=(id,3,3)

COUNT1	41	(X,4,Y): To COUNT2	(0,3,3)	if(isLeftLeader==T): to COUNT2
				if(isRightLeader==T): to WOG2

COUNT2	42	(0,4,X): To COUNT3	(id-1,4,3)	-

COUNT3	43	(X,5,Y): To RET	-	if isLeftLeader==T): to WOG1
				if(isRightLeader==T): (0,5,3)
				if(isLeader==N): (0,4,3)

WOG1	44	(X,5,3): To RET	(100,4,3)	-
WOG2	45	(100,X,Y): To COUNT2	-	-
RETREAT	51	To middle of Arbiter A,B. 1st time:,To RET2. 2nd time: To WAIT	(0,5,3)	-
RETREAT2	52	Move out of Results Box	-	-

specific message to modify its state, or else continue with its ongoing operation. In the rest of the section, we will describe the major stages of the algorithm according to Table II. Additionally, to see a summary of state transitions, one can refer to Tables II-VI.

A. The Gathering Stage

The Auto-Decrease-ID Gathering algorithm gathers Digit KBs that can represent up to two digits for each of the two operands in elementary mathematical operations. The algorithm starts by setting the Digit KBs in the WAIT status. Selectors then send ID messages that represent the operand digits. For example, Selectors may send a (5, 1, 4) message, where 5 is the operand, 1 represent the system Gathering stage, and the 4 tells that the sender is a Selector. Upon reception Digit KBs transition to the GATHER state. Among all Digit KBs in an Operand box, the first Digit KB to receive the ID message will perform three operations: (i) be assigned as the leader of the queue (Table VI), (ii) change its state to GATHER2, and (iii) send an (id-1, 1, 3) message to the next Digit KB.

The Digit KB who get an id from Selector KB. In this example, when a Digit KB get the (5, 1, 4) message, it will set its id to 5 and the variable isLeader to TRUE. Then it will send another gathering message (id-1, 1, 3), which means a Digit KB send a gathering message with data byte equals to 4. This is how the message codes work.

Digit KBs in GATHER2 wait for a message of (0, 1, X) and keep sending (id-1, 1, 3). If (ID-1) is greater than zero, the next Digit KB will change to GATHER2 and get an ID. The gathering process continues until ID decrements to 1. The next Digit KB in GATHER1 isn't activated and the previous KB in GATHER2 changes to state GATHER3. The message then flows from the queue's tail to the Selector. Upon reception, the latter sends a GO signal that helps Digits to transition to MOVE.

B. The Navigation Stage

Prior to the GO message, Arbiter KBs are in the FORWARD1 state and act as communication bridges between Digit KBs. During the FORWARD1 state, Arbiters simply relay ID messages down the operand queues. However, when Arbiters receive the GO signal, they change to the NAVIGATE1 state and begin emitting distance signals. At the intersection of the Arbiter's Navigate state and the Digit's GO state, these two KBs form a connection based on an ID-type to assist in the localization of the Digit robots. Both Arbiters and Digits can take on two classifications: type A bots or type B bots. A-type Digits will simply orbit around A-type Arbiters, and B-type digits will do so with B-type navigators. Furthermore, Digit KBs will constantly monitor the distance between itself and the closest Arbiters. A pre-designed STOP location for each KB in the Results Box in the TB is assigned such that when Digit KBs reach that position, they stop and transition to the STOP state. In the STOP state, Digit KBs synchronize their stop position. Leading Digit KBs send an (id, 3, 3) message out and non-leading Digits analyze the (X, 3, Y) message. If $X == 0$, they change their state to COUNT1. This continues until all active Digit KBs have switched to the COUNT1 state.

C. The Counting Stage

When the desired Digit KBs are centered in the Results box, a visual counting sequence begins to help children visualize the operations result. The counting process will have Digit KBs both bleep and blink their LED's starting the the left operand group and then continuing with the right operand group. Also, in this description the CARRY stage is not presented, so after the COUNTING stage, the RETREAT stage will proceed.

In the COUNT state, there are three sub-states: COUNT1&2&3 and two WAIT_OTHER_GROUP states. The process begins with the left operand bots. In COUNT1, all non-lead Digits must exit the STOP state. They are waiting for an (X, 4, Y) message before they transition. The Digit KB leader is the first to accept this message, which in turn makes the robot count by beeping and blinking. Afterwards the KB transition to COUNT2 and sends an ID-1 message to the rest of the active Digits in COUNT1. In the next time-step, the leading Digit also sends a Count End signal to active Digits and transition to COUNT3. When all left Digits are reach COUNT3, the left leader changes to WAIT_OTHER_GROUP1 and blocks all processes by sending a unique data byte message (100, 4, 3) except for a retreat message (X, 5, Y). This waiting period is also useful for children to count the resulting digits in the Results box.

Meanwhile, the right operand box is blocked in state COUNT1. The right leader will automatically change state to WAITOTHERGROUOP2, which waits for a (100, X, Y) message, after which it begins the count sequence and bliking and moves to COUNT2. Here too, non-leading Digits count like the ones in the left group. The the last Digit KB sends an (0, 4, 3) message driving the right group to COUNT3. The right leader then sends an (0, 5, 3) message to start the retreating process. The retreat messages is transmitted to the left operand box KBs through Messengers.

D. The Retreat Stage

The next state is the Retreat strategy. The crucial point of this state is to determine *where* to stop the KBs. As the Digit KBs orbit about the Arbiters, the Digits can compute a pair of distance measurements (through their light intensity): one comes from the A-type or B-type original pairing and the other is from an established adjacent Arbiter.

Given a pair of distance value, we cant know how much time it will take for the KB to reach the STOP distance threshold. For example, if we consider the distance between the Digit and the farthest Arbiter to be less than 51mm (a key distance in detecting stop points), the Digit KB wont always stop correctly. That particular distance may occur numerous times. A second threshold is required and one that leads to state RETREATE2. The condition to change to RETREATE2 is for the distance to the distant Arbiter to be less than 51mm. On the other hand, the condition to change from state RETREATE2 back to RETREAT1 is that the distance to the distant Arbiter is greater than 60mm. The latter will drive the Digit KB out of the critical boundary value and judge the 51mm threshold again. The Digit KBs will stop when they match the threshold the second time. When the Digit KBs have reached their original position, they will transition back to the WAIT state and the calculation is finished.

V. EXPERIMENTS

Kb robots where simulated in the V-REP environment. The V-REP simulator is a flexible and general purpose simulator with an integrated development environment. V-REP uses a distributed control architecture, where each object can be controlled via embedded scripts, plugins, ROS nodes, remote API clients, or custom solutions. Furthermore, controllers can be written in C/C++, Python, Java, Lua, Matlab, Octave or Urbi [20].

To test the basic functionality of the system, four addition operations were implemented, some of which required single-digit results and some which required double-digit results. We also varied the total KB number used in the TB, to evaluate the performance. The first two additions used a total of 43 KBs, the third demo, we use the same operation but reduce the total number of kilobots to 23. Finally, in the last operation we increase the number of KBs to an intermediate number to a total of 27 KBs. As part of our monitoring, we recorded four sets of time: the total duration of the operation, the selection process duration, the counting process duration, and the retreating process duration.

Addition 1: $9+7=16$

The first operation used 9 Digits and 10 Arbiters in the Left Operand Box, and 7 Digits and 8 Arbiters in the Right Operand Box. As well as 2 Selectors, 2 Messengers, 1 Operand robot (not visualized), and 4 passive Kbs for a total number of 43 KBs (see Fig. 2). The operation took a total of 413 seconds (Selecting Time: 16 secs, Counting Time: 42 secs, Moving Time: 320 secs). Fig 2 depicts a sequence of snap shots of the operation.

Addition 2: $3+5=8$

The second operation used 3 Digits and 4 Arbiters in the Left Operand Box, and 5 Digits and 6 Arbiters in the Right Operand Box. As well as 2 Selectors, 2 Messengers, 1 Operand robot (not visualized), and 20 passive Kbs for a total number of 43 KBs. The total duration for the operation was 336 seconds (Selecting Time: 21 secs, Counting Time: 19 secs, Moving Time: 280 secs).

Addition 3: $3+5=8$

In this third addition, we modify the total number of KBs used and reduce that number to 23. By eliminating passive KBs from the framework, all time durations diminished. The total duration for the operation was 162 seconds (Selecting Time: 11 secs, Counting Time: 9 secs, Moving Time: 140 secs).

Addition 4: $2+2=4$

The last operation, increase the number of KBs to an intermediate range and tried an addition operation with fewer number of Digit KBs, namely $2+2=4$. This operation used 2 Digits and 3 Arbiters in both Operand Boxes along with 2

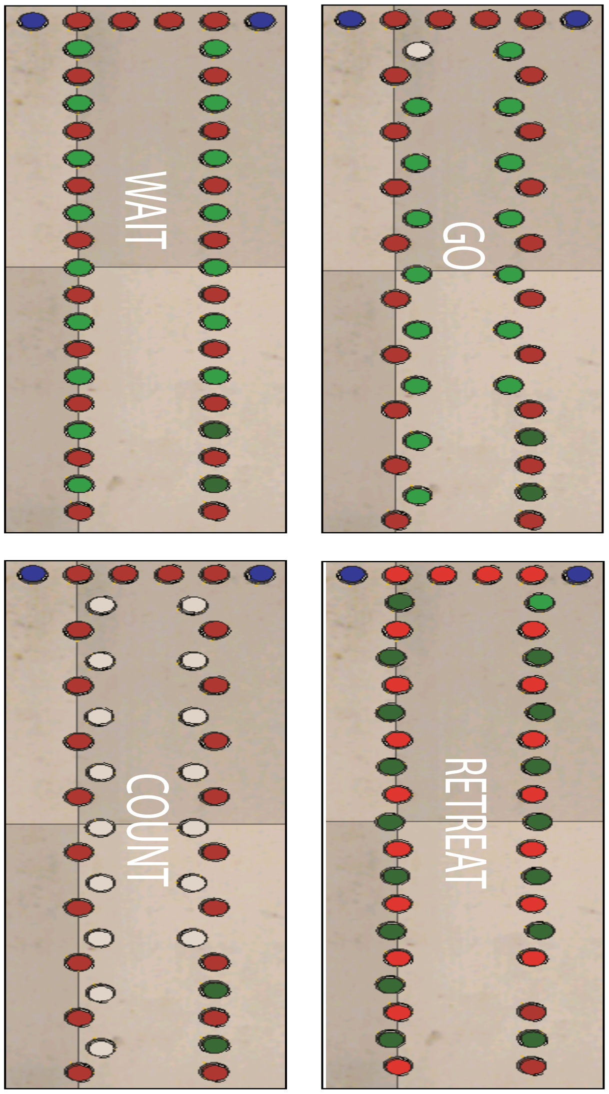

Fig. 2. The above figure shows the WAIT, GO, COUNT, and RETREAT stages of the addition 9+7=16 operation. The left operand box mobilizes 9 Digit KBs, the right operand box mobilizes 7 Digit KBs. The Results Box contains 16 Digit KBs that count off with blinking and beeping.

Selectors, 2 Messengers, 1 Operand robot (not visualized), and 12 passive Kbs for a total number of 27 KBs. The total duration for the operation was 189 seconds (Selecting Time: 5 secs, Counting Time: 3 secs, Moving Time: 180 secs).

In effect, operations with fewer digits compute their Selection process and Counting process more quickly. However these two durations are negligible compared to the Moving Time process. In simulation, the latter is dominated by the total number of KBs located in the operand boxes. There is a linear relation between the KB number and the total duration. When we reduced the total number of KBs from 43 to 23 for the $3 + 5 = 8$ operation, the total duration diminished roughly 50% as well: from 336 secs to 162 secs.

VI. CONCLUSION

This work presented a swarm framework to teach elementary mathematical addition operations. The Kilobot robot was selected as it's size and functionality also meet absorbability. A state-based synchronous algorithm was used to enable robots to represent the left and right operand digits in an operation and then navigate to represent Result digits. The system contemplates factors to engage young students and enhance a child's learning factors.

REFERENCES

[1] J. Edwards, "Next-generation robots offer sophisticated mobility, manipulation, and sensing capabilities [special reports]," *Signal Processing Magazine, IEEE*, vol. 30, no. 5, pp. 11–13, 2013.

[2] M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera, and R. Nagpal, "Kilobot: A low cost robot with scalable operations designed for collective behaviors," *Robotics and Autonomous Systems*, vol. 62, no. 7, pp. 966–975, 2014.

[3] R. Platt, F. Permenter, and J. Pfeiffer, "Using bayesian filtering to localize flexible materials during manipulation," *IEEE Transaction on Robotics*, vol. 27, no. 3, pp. 586–598, 2011.

[4] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3293–3298.

[5] D. Miner, "Swarm robotics algorithms: A survey," *Report, MAPLE lab, University of Maryland*, 2007.

[6] M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal, "Collective transport of complex objects by simple robots: theory and experiments," in *The 2013 Intl Conf on Aut Agents and Multi-Agent Systems*, 2013, pp. 47–54.

[7] W. v. Holland and W. F. Bronsvoort, "Assembly features in modeling and planning," *Robotics and computer-integrated manufacturing*, vol. 16, no. 4, pp. 277–294, 2000.

[8] K. C. Galloway, R. Jois, and M. Yim, "Factory floor: A robotically reconfigurable construction platform," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2467–2472.

[9] J. A. Rothermich, M. İ. Ecemiş, and P. Gaudiano, "Distributed localization and mapping with a robotic swarm," in *Swarm Robotics*. Springer, 2005, pp. 58–69.

[10] J. McLurkin and J. Smith, "Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots," in *Distributed Autonomous Robotic Systems 6*. Springer, 2007, pp. 399–408.

[11] S. Hettiarachchi and W. M. Spears, "Moving swarm formations through obstacle fields." in *IC-AI*, 2005, pp. 97–103.

[12] S. Papert, *The children's machine: Rethinking school in the age of the computer*. Basic Books, 1993.

[13] M. Resnick, S. Ocko, *et al.*, *LEGO/logo–learning through and about design*. Epistemology and Learning Group, MIT Media Laboratory, 1990.

[14] D. Rus, "Teaching robotics everywhere," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 1, pp. 15–94, 2006.

[15] N. Correll, R. Wing, and D. Coleman, "A one-year introductory robotics curriculum for computer science upperclassmen," *Education, IEEE Transactions on*, vol. 56, no. 1, pp. 54–60, 2013.

[16] N. C. Lye, K. W. Wong, and A. Chiou, "Framework for educational robotics: a multiphase approach to enhance user learning in a competitive arena," in *Edutainment Technologies. Educational Games and Virtual Reality/Augmented Reality Applications*. Springer, 2011, pp. 317–325.

[17] G. Keren, A. Ben-David, and M. Fridin, "Kindergarten assistive robotics (kar) as a tool for spatial cognition development in pre-school education," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1084–1089.

[18] M. Fridin, "Storytelling by a kindergarten social assistive robot: A tool for constructive learning in preschool education," *Computers & Education*, vol. 70, pp. 53–64, 2014.

[19] A. Chiou, "Teaching technology using educational robotics," in *Proceedings of The Australian Conference on Science and Mathematics Education (formerly UniServe Science Conference)*, vol. 10, 2012.

[20] M. F. E. Rohmer, S. P. N. Singh, "V-rep: a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.