




4-2017

Pythagorean Approximations for LEGO: Merging Educational Robot Construction with Programming and Data Analysis

Ronald I. Greenberg

Loyola University Chicago, Rgreen@luc.edu

Follow this and additional works at: https://ecommons.luc.edu/cs_facpubs

 Part of the [Computer Sciences Commons](#), [Geometry and Topology Commons](#), [Robotics Commons](#),
and the [Science and Mathematics Education Commons](#)

Author Manuscript

This is a pre-publication author manuscript of the final, published article.

Recommended Citation

Ronald I. Greenberg. Pythagorean approximations for LEGO: Merging educational robot construction with programming and data analysis. In Proceedings of the 8th International Conference on Robotics in Education, RiE 2017, volume 630 of Advances in Intelligent Systems and Computing, pages 65–76. Springer-Verlag, April 2017.

This Article is brought to you for free and open access by the Faculty Publications and Other Works by Department at Loyola eCommons. It has been accepted for inclusion in Computer Science: Faculty Publications and Other Works by an authorized administrator of Loyola eCommons. For more information, please contact ecommons@luc.edu.



This work is licensed under a [Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 License](#).

Pythagorean Approximations for LEGO: Merging Educational Robot Construction with Programming and Data Analysis*

Ronald I. Greenberg**

Loyola University, Chicago, IL 60626, USA,
rig@cs.luc.edu,

Abstract. This paper can be used in two ways. It can provide reference information for incorporating diagonal elements (for bracing or gear meshing) in educational robots built from standard LEGO® kits. Alternatively, it can be used as the basis for an assignment for high school or college students to recreate this information; in the process, students will exercise skills in both computer programming and data analysis. Using the paper in the second way can be an excellent integrative experience to add to an existing course; for example, the Exploring Computer Science high school curriculum concludes with the units “Introduction to Programming”, “Computing and Data Analysis”, and “Robotics”.

Keywords: computer science education, robotics, computer programming, data analysis

1 Introduction

Providing students with robotics experiences has become a popular and successful mechanism for broadening participation in computing and STEM more generally, retaining more students in these fields, and improving their learning. Robotics videos were found to be the most popular component of a series of brief computing outreach visits [12], and many studies have reported on successful programs using robots built from LEGO pieces, e.g. [5, 7, 2, 14].

Nonetheless, advice for students on how to build robots with LEGO kits is fairly limited, perhaps because LEGO is assumed to be a familiar medium from youthful play. An exceptional primer has been provided by Fred Martin [11]; in particular, it includes valuable tips regarding use of gears. It remains challenging, however, to incorporate any diagonal elements as opposed to placing each part across one dimension of an underlying regular 3D grid. To mesh gears along a diagonal or construct diagonal bracing, Martin suggests experimentation, though he notes the obvious applicability of the Pythagorean Theorem. In practice, such

* An abbreviated preliminary version of this paper appeared as [6]

** Supported in part by National Science Foundation grants CNS-1542971 and CNS-1543217.

experimentation with several cycles of building, disassembling, and rebuilding can be very frustrating to youthful robot builders.

This paper provides a more systematic approach to applying the Pythagorean Theorem to robot building. We use a spreadsheet to organize Pythagorean combinations that are close to exact, either with standard integral lengths or with some tricks that can be employed to place parts at spacings of half units (or sometimes even quarter units). Furthermore, the paper shows how to construct the spreadsheet using a simple program, and creating such a program and working with the resulting spreadsheet can be a good exercise in programming and data analysis to assign to students. In this way, students are motivated to complete the programming and data analysis due to its practical application to robot building tasks. This may form a particularly nice integration of the programming, data analysis, and robotics units of the Exploring Computer Science (ECS) curriculum [4], which has been implemented in many high schools in the United States. Initiated in Los Angeles, ECS is now in the seven largest US school districts and many other locations [3], and it has had a particularly strong impact in Chicago [1, 13]; it is also attracting some international attention.

2 LEGO Basics

One popular LEGO-based educational robotics program is Botball® [9], which has spread to many locations on four continents [10] and has major culminating tournaments/conferences in both North America (GCER) and Europe (ECER). Participating teams all work with a standard kit of LEGO and other parts assembled from those featured on the Botball web site [8]. Also, extremely widely used is the LEGO MINDSTORMS® kit [17]; for example, this was the primary recommended platform for the final “Robotics” unit of the Exploring Computer Science curriculum prior to the advent of other very inexpensive robots. The base kit for LEGO MINDSTORMS is similar to the Botball kit, though somewhat smaller, and various LEGO extension kits, available from a variety of vendors, also provide parts that are included in the Botball kit. While the MINDSTORMS kit refers to a package of “LEGO Bricks”, there are actually few if any traditional LEGO brick pieces in any of the standard robotics kits; most of the pieces can be ordered directly from `lego.com` by searching for “Technic” in the brick name [16], with some parts seeming to require a visit to the service portion of the site [15]. The main workhorse pieces are referred to as “beams” at `lego.com` or “liftarms” in Botball parts lists; most are completely straight or include a 90° angle. The width and height of a straight LEGO liftarm, as well as the space between adjacent holes along the length of a liftarm, are all one LEGO unit, which is 8mm. Various connectors are available for linking these pieces together.

With standard LEGO robotics parts, the most straightforward approach is to place all pieces along straight lines in the underlying 3D integer grid. This paper initially restricts all parts to be anchored to this integer grid while also considering diagonal components. Later, we also consider some tricks that can yield spacings at fractional LEGO units. Martin also suggests some constructions

involving fractional LEGO units, but these spacings are primarily achieved by using traditional LEGO bricks, which we have noted are not generally very available in current LEGO robotics kits.

Botball names are henceforth preferred, but we also provide information for looking up parts at lego.com. The standard gears are listed in Table 1.

Table 1. Standard gears in the Botball Lego kit.

Botball Description	Botball Quantity	LEGO design #	lego.com Name	Radius in LEGO Units
8 Tooth	12	3647	GEAR WHEEL T=8, M=1	0.5
16 Tooth	6	4019	GEAR WHEEL Z=16, M=1	1
24 Tooth	8	3648	GEAR WHEEL Z24	1.5
40 Tooth	6	3649	GEAR WHEEL 40T	2.5

In a typical alignment of gears along a liftarm, the gears of radius 1 will only mesh with each other (at a distance of 2), while the other gears may be abutted to one another in various combinations at distances of 1, 2, 3, 4, or 5. At any given time, we will focus on an arbitrary two dimensions of the underlying 3D grid and explore ways of achieving diagonal placement of liftarms and/or gears, while most components run either horizontally or vertically. Employing diagonal liftarms may be a useful bracing mechanism that uses fewer liftarms, or, of potentially greater value, a mechanism to align gears (and transfer motion) along a diagonal. (At least one team in the 2016 Botball competition sought such a design to use gearing and place wheels below and in front of the motor mounts in a standard metal chassis. These design elements were intended to allow the robot to ascend a ramp and straddle certain pieces on the game board.) For diagonals of length at most 5, it may also be possible to place gear centers on the rectilinear grid but with the gears meeting along a diagonal.

3 Near-Integral Triples

We begin by exploring ways to construct diagonals that connect to the underlying integer grid. An appropriately constructed spreadsheet provides a convenient way to view near-integral Pythagorean triples that may be helpful in LEGO construction. For example, the macro in Fig. 1 initializes a Microsoft Excel® spreadsheet of relevant data, shown here with appropriate choices for integral lengths only ($FRAC = 1$ and $HYPRND = 1$).

Additional parameter choices in the code of Fig. 1 were to limit the lengths of all triangle sides to 14 (longest available with a single liftarm), and to limit the slope (long leg divided by second leg) to 5, since constructions with slopes closer to 1 are of greater interest here as opposed to essentially running along a horizontal or vertical. In addition to showing leg lengths, slope, and hypotenuse length, the spreadsheet shows the approximate hypotenuse length (rounded to

the nearest multiple of $1/HYPRND$), the error relative to the actual hypotenuse length, and the absolute value of the error.

```

Sub Initialize()
Dim MAXLEG, MAXHYP, FRAC, HYPRND As Integer
MAXLEG = 14 'Maximum allowed leg length
MAXHYP = 14 'Maximum allowed hypotenuse length
MAXSLOPE = 5 'Maximum allowed slope (2nd leg divided by short leg)
FRAC = 1 'Fractions of Lego units allowed
      '(1 for whole units only, 2 for halves, 4 for quarters, etc.)
HYPRND = 1 'Approximate hypotenuse by rounding to specified fraction
      '(1 for whole units, 2 for halves, 4 for quarters, etc.)
Cells.Clear
Cells(1, 1).Value = "Short Leg" 'A1
Cells(1, 2).Value = "2nd Leg" 'B1
Cells(1, 3).Value = "Hypotenuse" 'C1
Cells(1, 4).Value = "Approx Hyp" 'D1
Cells(1, 5).Value = "Error" 'E1
Cells(1, 6).Value = "Abs Err" 'F1
Cells(1, 7).Value = "Slope" 'G1
Dim shortstep, secondstep As Integer 'Counters for short, 2nd leg lengths
Dim row As Integer: row = 2 'Counter starting at 1st row after headings
For shortstep = 1 To MAXLEG * FRAC
  For secondstep = shortstep To MAXLEG * FRAC
    Cells(row, 1).Value = shortstep/FRAC 'A row
    Cells(row, 2).Value = secondstep/FRAC 'B row
    Cells(row, 3).Formula = "=SQRT(A" & row & "^2 B" & row & "^2)" 'C row
    Cells(row, 4).Formula = "=MROUND(C" & row & ", " & 1/HYPRND & ")" 'D row
    Cells(row, 5).Formula = "=D" & row & "-C" & row 'E row
    Cells(row, 6).Formula = "=ABS(E" & row & ")" 'F row
    Cells(row, 7).Formula = "=B" & row & "/A" & row 'G row
    If Cells(row, 4)<=MAXHYP And Cells(row, 7)<=MAXSLOPE Then row = row + 1
  Next secondstep
Next shortstep
Rows(row).EntireRow.Delete 'Remove last row failing condition at loop end
End Sub

```

Fig. 1. A Visual Basic for Applications macro to initialize a Microsoft Excel spreadsheet of Pythagorean triple data.

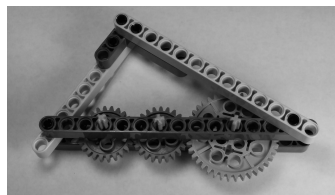
Constructing a macro and spreadsheet of this sort is a feasible exercise for beginning computing students; the most difficult programming concept involved is nested for-loops. The other rudiments of working with VBA in Excel can be easily conveyed to students. (The task also can be simplified somewhat by generating only data values rather than formulas in all the cells.) This exercise can even be completed in Scratch (<http://scratch.mit.edu>), the environment

regularly used for programming in the Exploring Computer Science curriculum and many other beginning computing classrooms. In the Scratch version, one can generate rows of comma-separated values as items in a list, right click on the list to export to a file, and then read the file into Excel.

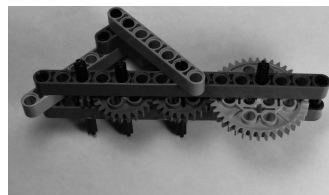
The macro of Fig. 1 generates a spreadsheet with 68 data rows. Sorting by absolute error, we find 11 rows with absolute error of less than 0.1 LEGO units as shown in Table 2. (For presentational convenience, the values here and in further figures are rounded to a small number of decimal places; this also helps provide opportunities to ask for something more on a homework assignment even if students find this paper.) It is actually possible to construct nearly all of the 68 possibilities as verified by constructing cases near the bottom of the sorted spreadsheet, but the ones listed in Table 2 involve less deformation of the pieces. As an example, Fig. 2(a) shows the case with least nonzero absolute error (7-11-13) used to hold a sequence of gears. (At least one combination remains unworkable, e.g., trying to build a 1-1-1 using a bent liftarm for the two legs.)

Table 2. The 11 near-integral (or integral) Pythagorean triples with sides ≤ 14 with least absolute error.

Short Leg	2nd Leg	Hypotenuse	Approx. Hyp.	Error	Abs. Error	Slope
3	4	5	5	0	0	1.333
5	12	13	13	0	0	2.4
6	8	10	10	0	0	1.333
7	11	13.038	13	-0.038	0.038	1.571
8	9	12.042	12	-0.042	0.042	1.125
4	8	8.944	9	0.056	0.056	2
4	7	8.062	8	-0.062	0.062	1.75
5	5	7.071	7	-0.071	0.071	1
5	13	13.928	14	0.072	0.072	2.6
5	11	12.083	12	-0.083	0.083	2.2
1	5	5.099	5	-0.099	0.099	5



(a)



(b)

Fig. 2. Two approximate Pythagorean triples with sides ≤ 14 . **(a)** The one with least absolute error, (7-11-13). **(b)** The one with least absolute error for a hypotenuse slope of 1, (5-5-7). The hypotenuse in this case is extended to provide more space for gears.

Probably, the most useful way to use the spreadsheet is to sort by slope after sorting by absolute error. Then when a particular slope is desired along which to align a sequence of gears, one may select the corresponding option with least absolute error. Table 3 shows the triples with least absolute error for slopes from 1 to 5 at intervals of 0.5: As an example of this approach, the tightest construction of slope 1 is a 5-5-7. These dimensions can be used to create a tight construction while actually extending the hypotenuse farther to hold a longer sequence of gears; for example, see Fig. 2(b).

Table 3. The approximate Pythagorean triples with sides ≤ 14 with least absolute error for slopes of 1 to 5 at intervals of 0.5.

Short Leg	2nd Leg	Hypotenuse	Approx. Hyp.	Error	Abs. Error	Slope
5	5	7.071	7	-0.071	0.071	1
6	9	10.817	11	0.183	0.183	1.5
4	8	8.944	9	0.056	0.056	2
4	10	10.770	11	0.230	0.230	2.5
1	3	3.162	3	-0.162	0.162	3
2	7	7.280	7	-0.280	0.280	3.5
1	4	4.123	4	-0.123	0.123	4
2	9	9.220	9	-0.220	0.220	4.5
1	5	5.099	5	-0.099	0.099	5

The example constructions of Fig. 2 use liftarms along the hypotenuse to hold gears securely in place, but one can also consider placing liftarms horizontally and vertically only while attempting to place two gear centers on such liftarms so that the gears mesh diagonally along the hypotenuse. In this case, the hypotenuse can be no longer than 5, the largest possible separation of adjacent gear centers (using gears of radius 2.5). The most obvious possibility is to use the exact Pythagorean triple 3-4-5. Even this exact triple may leave the gears able to slip past one another due to the possibility of axles wobbling in liftarm holes, etc., but with good bracing, it seems to be feasible also to work with the three triples of small hypotenuse that come next in absolute error: 1-5-5, 1-4-4, or even 1-3-3. Beyond this, the error switches sign and the gears bind, or the error is too great to ensure that the gears do not slip. Figure 3 shows the most extreme pair of cases from the four just mentioned.

(One may also note that the “ 1×11.5 Liftarm Double Bent” pieces used in Fig. 3(a) can be used to attach liftarms at a slope of 1. Additionally, the “ 1×9 Liftarm Bent (3×7)” and the “ 1×7 Liftarm Bent (4×4)” form an angle 90° greater than the small angle of a 3-4-5 triangle.)



Fig. 3. (a) The exact Pythagorean triple 3-4-5 with the hypotenuse formed solely by gears. (b) the approximate Pythagorean triple 1-3-3 that has the greatest absolute error (0.16) that seems workable in practice.

4 Half-Unit Spacing

The left-hand side of Fig. 4 shows a simple construction using a Cam and a “1 × 3 Liftarm Thin” that can extend any ordinary liftarm by 1.5 units (to the leftmost axle center). Looking also at the right-hand side of Fig. 4, we can see that cams, thin lift arms (including “Triangle pieces”), the “Bush 1/2” and “Nut 8-32 Keeps (black)” all can be used to create half-unit spacing perpendicular to a liftarm. All these parts are provided in the standard Botball kit, and all but the nut (and accompanying screws) are available at lego.com; see Table 4.

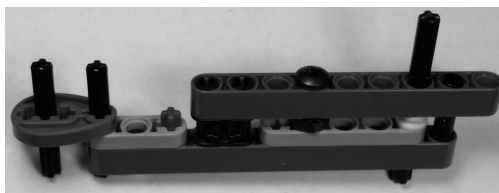


Fig. 4. An illustration of various ways to incorporate half-unit spacing along the length of a liftarm or in a perpendicular direction.

Tweaking our macro (Fig. 1) for half-unit spacings up to 15.5 (longest liftarm extended by 1.5 units as in Fig. 4) by setting $MAXLEG = 15.5$, $MAXHYP = 15.5$, $FRAC = 2$ and $HYPKND=2$, we obtain 301 data rows if we eliminate sides of length just 0.5. Extracting the triples with least absolute error for slopes from 1 to 5 at intervals of 0.5, we obtain Table 5. All but the last of these is new in comparison to Table 3.

As in Sect. 3, we can again consider using only gears (no liftarm) along the hypotenuse. We already know standard gear pairings achieve gear center spacings of 1, 2, 3, 4, or 5, so we can now consider an expanded set of triples with integral hypotenuse from 1 to 5. Note that we can also achieve half-unit gear separations

Table 4. Parts in the Botball Lego kit that are useful for half-unit spacings.

Botball Description	Botball Quantity	LEGO design #	lego.com Name
Bush 1/2	42	32123	1/2 BUSH
Nut 8-32 Keeps (black)	85	—	—
1 × 3 Liftarm Thin Triangle	16	6632	TECHNIC LEVER 3M
	4	2905	TRIANGLE
Cam	4	6575	COMB WHEEL

Table 5. The approximate Pythagorean triples with sides ≤ 15.5 with least absolute error for slopes of 1 to 5 at intervals of 0.5 when half-unit side lengths are allowed.

Short Leg	2nd Leg	Hypotenuse	Approx. Hyp.	Error	Abs. Error	Slope
6	6	8.485	8.5	0.015	0.015	1
5	7.5	9.014	9	-0.014	0.014	1.5
2	4	4.472	4.5	0.027	0.027	2
5	12.5	13.463	13.5	0.037	0.037	2.5
3	9	9.487	9.5	0.013	0.013	3
4	14	14.560	14.5	-0.060	0.060	3.5
3.5	14	14.431	14.5	0.069	0.069	4
1	4.5	4.610	4.5	-0.110	0.110	4.5
1	5	5.099	5	-0.099	0.099	5

of 1.5, 2.5, or 3.5 by combining the “16 Tooth” gear in Fig. 1 with the other standard gears. We also can use a new trick to place gears with center spacings in half-units; specifically, the double bevel (DB) gears, though recommended by Martin [11] only for changing the axis of rotation, can mesh like traditional gears do as long as we allow new gear center spacings. The three types of double bevel gears shown in Table 6, can mesh with one another at distances of 1.5, 2, 2.5, 3, 3.5, or 4.5. Considering these and the traditional gear spacings as possible hypotenuse values yields 32 data rows. Table 7 shows the 20 rows with the least absolute error, ending with the familiar 1-3-3 considered at the end of Sect. 3, beyond which the error seems too high to be prudent.

Table 6. Standard double bevel gears in the Botball Lego kit.

Botball Description	Botball Quantity	LEGO design #	lego.com Name	Radius in LEGO Units
12 Tooth DB	4	32270	DOUBLE CONICAL WHEEL Z12,1M	0.75
20 Tooth DB	8	32269	DOUBLE CONICAL WHEEL Z20,1M	1.25
36 Tooth DB	6	32498	DOUBLE CONICAL WHEEL Z36	2.25

Table 7. The approximate Pythagorean triples with short hypotenuse with least absolute error while allowing half units.

Short Leg	2nd Leg	Hypotenuse	Approx. Hyp.	Error	Abs. Error	Slope
1.5	2	2.5	2.5	0	0	1.333
3	4	5	5	0	0	1.333
2	4	4.472	4.5	0.028	0.028	2
2	3.5	4.031	4	-0.031	0.031	1.75
2.5	2.5	3.536	3.5	-0.036	0.036	1
3.5	3.5	4.950	5	0.050	0.050	1
2	4.5	4.924	5	0.076	0.076	2.25
1.5	2.5	2.915	3	0.085	0.085	1.667
1	1	1.414	1.5	0.086	0.086	1
2.5	3	3.905	4	0.095	0.095	1.2
1	5	5.099	5	-0.099	0.099	5
2	3	3.606	3.5	-0.106	0.106	1.5
3	3.5	4.610	4.5	-0.110	0.110	1.167
1	4.5	4.610	4.5	-0.110	0.110	4.5
1.5	1.5	2.121	2	-0.121	0.121	1
1	4	4.123	4	-0.123	0.123	4
1	3.5	3.640	3.5	-0.140	0.140	3.5
1.5	3	3.354	3.5	0.146	0.146	2
2.5	4.5	5.148	5	-0.158	0.158	1.8
1	3	3.162	3	-0.162	0.162	3

5 Quarter-Unit Gear Spacing

Quarter-unit spacing is not generally easy to achieve, but we can consider using certain gear combinations to form a hypotenuse that measures in quarter units. Specifically, we can mesh a traditional gear (Table 1) with a double bevel gear (Table 6) to obtain a spacing of 1.25, 1.75, 2.25, 2.75, 3.25, 3.75, 4.25 or 4.75. Tweaking the macro of Fig. 1 as in the previous section but with $HYP\text{RND} = 4$, we obtain the approximate Pythagorean triples of Table 8 with a hypotenuse that can be formed by meshing a traditional gear with a double bevel gear.

6 Gear Ratios

It can also be helpful to generate a spreadsheet to keep track of the spacings and gear ratios that result from all the different gear pairings that can be considered. Spacings that are integral or in half-units can be used horizontally or vertically, or perhaps on a diagonal per Table 7. Quarter-unit spacings can be utilized as per Table 8.

Generating a spreadsheet of gear ratios and spacings also can be assigned to students as an elementary exercise using nested for-loops. An extra programming concept that can be introduced here is creation of an abstract data type. For

Table 8. The approximate Pythagorean triples with a hypotenuse that can be formed by meshing a traditional gear with a double bevel gear in order of least absolute error.

Short Leg	2nd Leg	Hypotenuse	Approx. Hyp.	Error	Abs. Error	Slope
1.5	4.5	4.743	4.75	0.007	0.007	3
3	3	4.243	4.25	0.007	0.007	1
1	2	2.236	2.25	0.014	0.014	2
1.5	4	4.272	4.25	-0.022	0.022	2.667
2.5	4	4.717	4.75	0.033	0.033	1.6
2	2.5	3.202	3.25	0.049	0.049	1.25
2.5	3.5	4.301	4.25	-0.051	0.051	1.4
1	1.5	1.803	1.75	-0.053	0.053	1.5
1	2.5	2.693	2.75	0.057	0.057	2.5
1.5	3.5	3.808	3.75	-0.058	0.058	2.333
2	2	2.828	2.75	-0.078	0.078	1
1	3	3.162	3.25	0.088	0.088	3
1.5	3	3.354	3.25	-0.104	0.104	2
1	3.5	3.640	3.75	0.110	0.110	3.5

example, in VBA, one may wish to fill an array of gears that will be paired, with each gear being of the following user-defined type:

```
Type gear
    name As String
    teeth As Integer
    radius As Single
End Type
```

It would also be straightforward to generate a CSV file using this approach in any number of other programming languages, though Scratch would not be a convenient environment for working with an abstract data type.

The data resulting from considering all the gear pairings is perhaps most interesting when sorted by spacing as in Table 9. This exposes a number of ways to build fixed spacing between gears and later tweak the gear ratio with minimal reconstruction, more so than considering the single possibility cited by Martin of being able to interchange a 24-tooth/8-tooth pair with a pair of 16-tooth gears.

7 Conclusion

The possible constructions indicated in this paper are by no means exhaustive. Various fractional spacings may also be achieved by interposing metal pieces, bricks, or plates/tiles between liftarms, but the constructions here use the more plentiful pieces in a standard Botball kit and retain a regular grid-based approach, either on the traditional grid or on a half-unit grid. There also are some other types of gears that may be available, for example, the single bevel, crown, and worm gears provided in the Botball kit, but this paper considers the most

Table 9. All possible pairings of the gears of Table 1 (40, 24, 16, and 8 tooth gears) and Table 6 (36, 20, and 12 tooth double bevel gears) with the resulting spacing and gear ratio, sorted by spacing.

Largest Gear Teeth	Second Gear Teeth	Gear Center Spacing in LEGO Units	Gear Ratio	Largest Gear Teeth	Second Gear Teeth	Gear Center Spacing in LEGO Units	Gear Ratio
40	40	5	1	36	8	2.75	4.5
40	36	4.75	1.111	20	16	2.75	1.25
36	36	4.5	1	24	16	2.5	1.5
40	20	4.25	2	20	12	2.5	1.667
40	24	4	1.667	24	12	2.25	2
36	20	4	1.8	20	8	2.25	2.5
36	24	3.75	1.5	24	8	2	3
40	16	3.5	2.5	16	16	2	1
20	20	3.5	1	16	12	1.75	1.333
40	12	3.25	3.333	16	8	1.5	2
36	16	3.25	2.25	12	12	1.5	1
24	20	3.25	1.2	12	8	1.25	1.5
40	8	3	5	8	8	1	1
36	12	3	3				
24	24	3	1				

straightforward usages of gears aligned in the same axis of rotation. The several reference tables in the paper should be useful to robot builders, and the methodological approach provides a good basis for assigning programming and data analysis tasks to students.

References

1. L. Dettori, R. I. Greenberg, S. McGee, and D. Reed. The impact of the Exploring Computer Science instructional model in Chicago Public Schools. *Computing in Science & Engineering (Special Issue: Best of RESPECT 2015)*, 18(2):10–17, March/April 2016.
2. B. Ericson and T. McKlin. Effective and sustainable computing summer camps. In *SIGCSE '12*, pages 289–294. Association for Computing Machinery, 2012.
3. Exploring Computer Science. A national program. <http://www.exploringcs.org/about/ecs-now>, 2016. Accessed Dec. 29, 2016.
4. J. Goode and G. Chapman. Exploring computer science (version 7.0). <http://www.exploringcs.org/curriculum>, 2016.
5. L. M. Grabowski and P. Brazier. Robots, recruitment and retention: Broadening participation through CS0. In *Proceedings of 2011 Frontiers in Education Conference (FIE)*, pages F4H1–5, 2011.
6. R. I. Greenberg. Pythagorean combinations for LEGO robot building. In *Proceedings of the Global Conference on Educational Robotics (GCER)*. KISS Institute for Practical Robotics, July 2016. http://files.kipr.org/gcer/2016/GCER_2016_Papers/Pythagorean_Combinations_for_Lego_Robot_Building.pdf.

7. S. H. Kim and J. W. Jeon. Introduction for freshmen to embedded systems using LEGO Mindstorms. *IEEE Transactions on Education*, 52(1):99–108, 2009.
8. KISS Institute for Practical Robotics. Products. <http://botballstore.org/products>. Accessed May 21, 2017.
9. KISS Institute for Practical Robotics. Botball® educational robotics program. <http://www.botball.org>, 2015. Accessed June 8, 2016.
10. KISS Institute for Practical Robotics. Regions & teams. <http://www.botball.org/regions-teams>, 2016. Accessed Jan. 2, 2017.
11. F. G. Martin. The art of LEGO design. *The Robotics Practitioner: The Journal for Robot Builders*, 1(2), Spring 1995.
12. S. McGee, R. I. Greenberg, D. F. Reed, and J. Duck. Evaluation of the IMPACTS computer science presentations. *The Journal for Computing Teachers*, pages 26–40, Summer 2013. International Society for Technology in Education, <http://www.iste.org/resources/product?id=2853>.
13. S. McGee, R. McGee-Tekula, J. Duck, R. I. Greenberg, L. Dettori, D. F. Reed, B. Wilkerson, D. Yanek, A. M. Rasmussen, and G. Chapman. Does a taste of computing increase computer science enrollment? *Computing in Science & Engineering (Special Issue: Best of RESPECT 2016)*, 9(3):8–18, April 2017.
14. R. B. Osborne, A. J. Thomas, and J. R. N. Forbes. Teaching with robots: A service learning approach to mentor training. In *SIGCSE '10*, pages 172–176. Association for Computing Machinery, 2010.
15. The LEGO Group. Bricks & pieces. <http://service.lego.com/replacementparts>. Accessed Aug. 15, 2016.
16. The LEGO Group. Pick a brick. <http://shop.lego.com/en-US/Pick-a-Brick>. Accessed May 21, 2017.
17. The LEGO Group. MINDSTORMS EV3. <http://www.lego.com/en-us/mindstorms/products>, 2016. Accessed July 11, 2016.