Команда «Стриж»

Багрова Ольга Алексеевна, учитель информатики и робототехники МБОУ СОШ г. Пионерского, Калининградская область - руководитель команды;
Багрова Ангелина Артуровна, учащаяся 9-го класса
Неговоров Константин Владимирович, учащийся 9-го класса

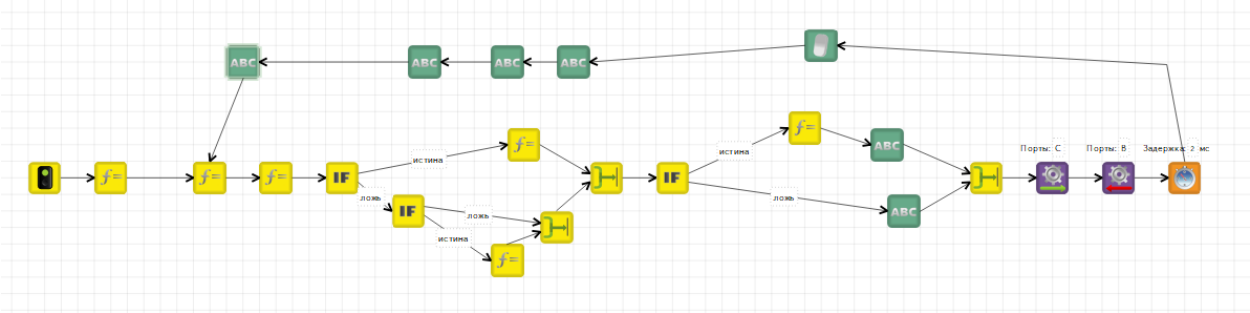
Описание робота

Робот собран на базе конструктора Lego Mindstorms EV3. В конструкции робота используются два малых мотора, два датчика света. Используется коронная передача для изменения плоскости вращения вала. Размеры робота: 21,5 см х 18 см. По своей форме робот напоминает равносторонний треугольник.

Программирование робота велось в TRIK Studio 2022.2. Программа робота запоминает, с какой стороны была линия, и в случае схождения с неё возвращает робота на траекторию движения. При тестировании робота убедились, что робот проезжает более пяти кругов, не сбиваясь с траектории.

Робот получил название «Стриж» из-за своей быстроты.

Программа



Скомпилированный код программы

```
// WARNING: Unreadable code! Close your eyes and run away!

// Don`t tell me I didn`t warn you. Now a little tips to understand this.
// This is an asm-like code to deal with native EV3 software. All your nice TRIK Studio
// mathematical one-line expressions are now big and slow pieces of ... code.
// For example "a = 100 * 200 - 50" will be translated into
//
//    MUL32(100, 100, _int_temp_result_1) /* _temp_result_* is something like asm register here */
//    SUB32(_int_temp_result_1, 50, _int_temp_result_2)
//    MOVE32_32(_temp_result_2, a)
//
```

```
// No temporary variables count optimization is performed for now, so suffer.
// Control flow is emulated with goto statements (kill me please).
// Note: 8, 16, 32-bit and float mathematical functions accept arguments only in corresponding bitness,
// so many temporary variables for type casting will be met there.

// Again, reading the code below may harm your mind. If something behaves strangely please contact developers.

	DATAF pi
	DATA32 _int_temp_result_1
	DATA32 _int_temp_result_2
	DATA32 _int_temp_result_3
	DATA32 dom
	DATA32 kPad
	DATA32 otk
	DATA32 speed
	DATA32 sr2
	DATA32 sr3
	DATA8 _bool_temp_result_1
	DATA8 _bool_temp_result_2
	DATA8 _bool_temp_result_3
	DATAF _float_temp_result_1
	DATAF _float_temp_result_2
	DATAF k
	DATAF yk
	DATAS _string_temp_result_1 255

vmthread MAIN
{
	MOVEF_F(3.141592653589793F, pi)

	DATA32 timer
	DATA8 _temp_sensor_value_8
	DATAF _temp_sensor_value_f



	a54b8d22f4631afab66fbfca4efdc:
	MOVEF_F(0.85F, k)
	MOVE32_32(0, dom)
	MOVE32_32(50, kPad)
	MOVE32_32(100, otk)
	MOVE32_32(95, speed)
```

```
e37ffc4d1982a9b4df98de0948:
INPUT_READ(0, 1, 0, 0, _temp_sensor_value_8)
MOVE8_32(_temp_sensor_value_8, _int_temp_result_1)
MOVE32_32(_int_temp_result_1, sr2)
INPUT_READ(0, 2, 0, 0, _temp_sensor_value_8)
MOVE8_32(_temp_sensor_value_8, _int_temp_result_2)
ADD32(_int_temp_result_2, 10, _int_temp_result_3)
MOVE32_32(_int_temp_result_3, sr3)


f02830cd4e769ce030170ac3dfb4:
SUB32(sr2, sr3, _int_temp_result_1)
MOVE32_F(_int_temp_result_1, _float_temp_result_2)
MULF(_float_temp_result_2, k, _float_temp_result_1)
MOVEF_F(_float_temp_result_1, yk)

eec334d18a74706824d8d82c2666635:
ADD32(sr3, 30, _int_temp_result_1)
CP_LT32(_int_temp_result_1, sr2, _bool_temp_result_1)
JR_FALSE(_bool_temp_result_1, b0fb2a30843ea94f70ad1dc08f13b)
    JR(ac0ff24d3427fb7478498a8a9a701)
b0fb2a30843ea94f70ad1dc08f13b:
    JR(bce7d2572ff4e1391fbe8e9284044f8)

ac0ff24d3427fb7478498a8a9a701:
MOVE32_32(1, dom)

c6ee39b73940db95e675e19e656e37:

da3ab3384521bd2725355b8ca708:
CP_GT32(sr2, kPad, _bool_temp_result_1)
CP_GT32(sr3, kPad, _bool_temp_result_2)
AND8(_bool_temp_result_1, _bool_temp_result_2, _bool_temp_result_3)
JR_FALSE(_bool_temp_result_3, e41b13ab9e48798a7ea1b8037a514a)
    JR(ce6afd5d188438580a0eff173dcb84d)
e41b13ab9e48798a7ea1b8037a514a:
    JR(b79a608ae744e38962fccd5bfe39ee3)

bce7d2572ff4e1391fbe8e9284044f8:
ADD32(sr2, 30, _int_temp_result_1)
CP_LT32(_int_temp_result_1, sr3, _bool_temp_result_1)
JR_FALSE(_bool_temp_result_1, a267b8c13baa49f2b053861a56fc6b04)
    JR(d537c374e4b90b6a441fb97e194)
a267b8c13baa49f2b053861a56fc6b04:
```

```
            JR(db64890ac966a113c169361)

        ce6afd5d188438580a0eff173dcb84d:
        MUL32(otk, dom, _int_temp_result_1)
        MOVE32_F(_int_temp_result_1, yk)

        ee6939913f246d8a726aaadafc3bad1:
        UI_DRAW(TEXT, FG_COLOR, 0, 20, '24352435')
        UI_DRAW(UPDATE)

        JR(cf659f063727494ead2b2c60ad66f1af)
        b79a608ae744e38962fccd5bfe39ee3:
        UI_DRAW(TEXT, FG_COLOR, 0, 20, '                    ')
        UI_DRAW(UPDATE)

        cf659f063727494ead2b2c60ad66f1af:

        b7f10eef0b44477986ae10f597b7642:
        MOVE32_F(speed, _float_temp_result_2)
        SUBF(_float_temp_result_2, yk, _float_temp_result_1)
        MOVEF_32(_float_temp_result_1, _int_temp_result_1)
        DATA32 d459d74709a0524b3dd94997c1
        MOVE32_32(_int_temp_result_1, d459d74709a0524b3dd94997c1)
        CALL(motors_overflow_check_EV3_KERNEL_util,
d459d74709a0524b3dd94997c1, d459d74709a0524b3dd94997c1)
        OUTPUT_POWER(0, 4, d459d74709a0524b3dd94997c1)
        OUTPUT_START(0, 4)

        cbe7130d0644d50b5140fea4fcd9533:
        MOVE32_F(speed, _float_temp_result_2)
        ADDF(_float_temp_result_2, yk, _float_temp_result_1)
        MOVEF_32(_float_temp_result_1, _int_temp_result_1)
        DATA32 a0814912854f3898e42681cce7cc6a
        MOVE32_32(_int_temp_result_1, a0814912854f3898e42681cce7cc6a)
        MUL32(a0814912854f3898e42681cce7cc6a, -1,
a0814912854f3898e42681cce7cc6a)
        CALL(motors_overflow_check_EV3_KERNEL_util,
a0814912854f3898e42681cce7cc6a, a0814912854f3898e42681cce7cc6a)
        OUTPUT_POWER(0, 2, a0814912854f3898e42681cce7cc6a)
        OUTPUT_START(0, 2)

        aa2aa693350844ce9593bc599f0bb974:
        TIMER_WAIT(2, timer)
        TIMER_READY(timer)
```

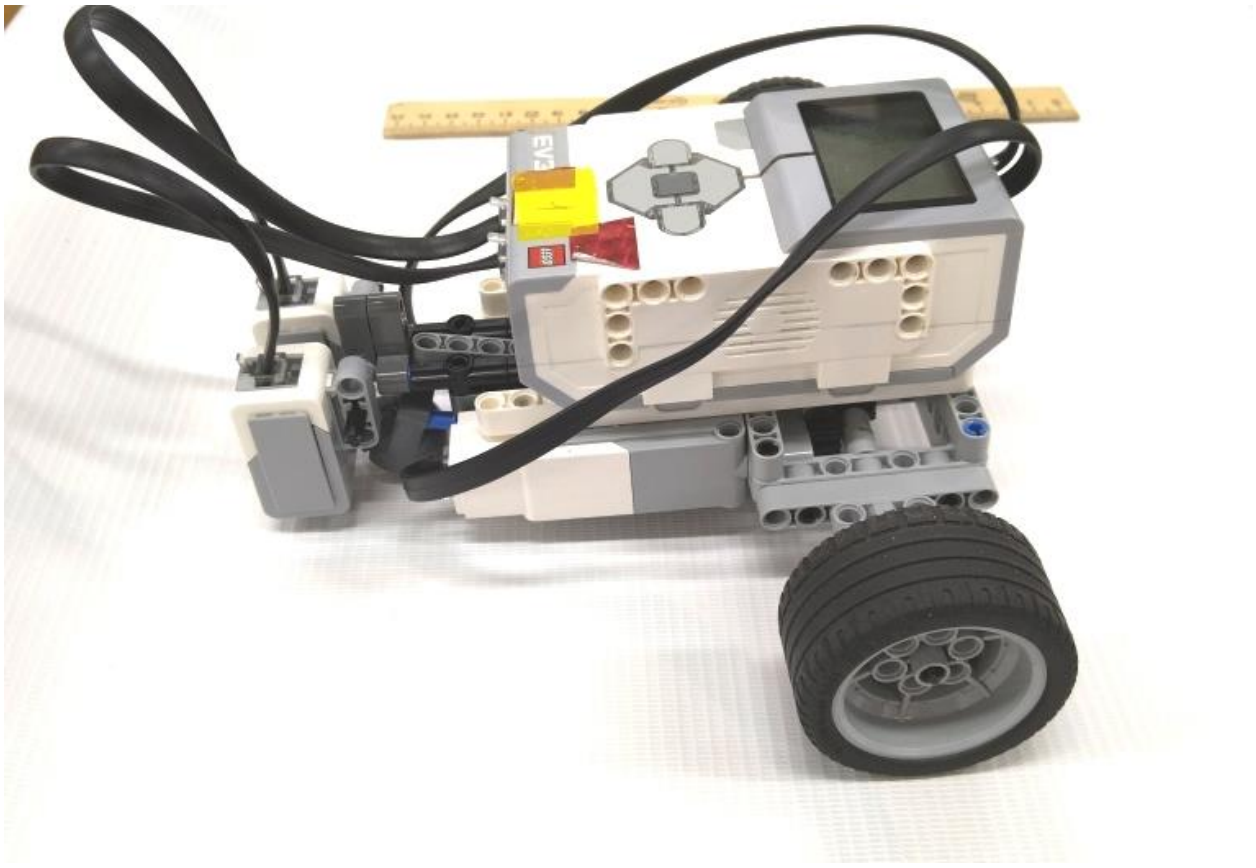
```
	aa10966eceee473c8d276cbf93328717:
	UI_DRAW(FILLWINDOW, 0, 0, 0)
	UI_DRAW(UPDATE)

	d456e05986e6451e8b79413003fc792e:
	STRINGS(NUMBER_FORMATTED, sr3, '%d', 10, _string_temp_result_1)
	UI_DRAW(TEXT, FG_COLOR, 100, 1, _string_temp_result_1)
	UI_DRAW(UPDATE)

	a8cdab54207bbf7d5c8fb039c2b:
	STRINGS(NUMBER_FORMATTED, sr2, '%d', 10, _string_temp_result_1)
	UI_DRAW(TEXT, FG_COLOR, 1, 1, _string_temp_result_1)
	UI_DRAW(UPDATE)

	dd0000fb4b7c8359adddc0a57510:
	STRINGS(NUMBER_FORMATTED, dom, '%d', 10,
_string_temp_result_1)
	UI_DRAW(TEXT, FG_COLOR, 0, 10, _string_temp_result_1)
	UI_DRAW(UPDATE)

	ffb8b8e881da482bb649fd6bc3315fb1:
	STRINGS(NUMBER_FORMATTED, otk, '%d', 10, _string_temp_result_1)
	UI_DRAW(TEXT, FG_COLOR, 100, 10, _string_temp_result_1)
	UI_DRAW(UPDATE)

	JR(e37ffc4d1982a9b4df98de0948)
	d537c374e4b90b6a441fb97e194:
	MOVE32_32(-1, _int_temp_result_1)
	MUL32(1, _int_temp_result_1, _int_temp_result_1)
	MOVE32_32(_int_temp_result_1, dom)

	JR(db64890ac966a113c169361)
	db64890ac966a113c169361:

	JR(c6ee39b73940db95e675e19e656e37)
	JR(c6ee39b73940db95e675e19e656e37)


__programEnd:
}

// utils functions block start
subcall motors_overflow_check_EV3_KERNEL_util
{
	IN_32 src
```

```
        OUT_32 dst

        MOVE32_32(src,dst)

        DATA32 lowerBound
        MOVE32_32(-100,lowerBound)
        DATA32 upperBound
        MOVE32_32(100,upperBound)

        JR_LT32(src, 0, lowThenZero)
        JR_LT32(src, upperBound, endLabel)
        MOVE32_32(upperBound,dst)
        JR(endLabel)

lowThenZero:
        JR_GTEQ32(src, lowerBound, endLabel)
        MOVE32_32(lowerBound,dst)

endLabel:
}

subcall clp2_EV3_KERNEL_util
{
        IN_32 src
        OUT_32 dst

        MOVE32_32(1, dst)
        JR_LTEQ32(src, 1, endLabel)

        DATA32 tmp
        DATA32 xRotated
        MOVE32_32(src, tmp)
        SUB32(tmp, 1, tmp)
        MOVE32_32(tmp, xRotated)
        DIV32(xRotated, 2, xRotated)
        OR32(tmp, xRotated, tmp)
        MOVE32_32(tmp, xRotated)
        DIV32(xRotated, 4, xRotated)
        OR32(tmp, xRotated, tmp)
        MOVE32_32(tmp, xRotated)
        DIV32(xRotated, 16, xRotated)
        OR32(tmp, xRotated, tmp)
        MOVE32_32(tmp, xRotated)
        DIV32(xRotated, 256, xRotated)
        OR32(tmp, xRotated, tmp)
```

```
        MOVE32_32(tmp, xRotated)
        DIV32(xRotated, 65536, xRotated)
        OR32(tmp, xRotated, tmp)

        ADD32(tmp, 1, dst)
endLabel:
}

subcall write32Array_EV3_KERNEL_util
{
        IN_32  array
        IN_32  pos
        IN_32  value

        DATA32 size
        ARRAY(SIZE, array, size)
        JR_LT32(pos, size, writing_label)
        DATA32 newSize
        CALL(clp2_EV3_KERNEL_util, size, newSize)
        ARRAY(RESIZE, array, newSize)

        writing_label:
        ARRAY_WRITE(array, pos, value)
}

subcall assign32Array_EV3_KERNEL_util
{
        IN_32 srcArray
        IN_32 dstArray

        DATA32 sizeSrc
        DATA32 sizeDst
        ARRAY(SIZE, srcArray, sizeSrc)
        ARRAY(SIZE, dstArray, sizeDst)
        JR_LTEQ32(sizeSrc, sizeDst, copy_label)
        ARRAY(RESIZE, dstArray, sizeSrc)

        copy_label:
        ARRAY(COPY, srcArray, dstArray)
}

subcall B2U32_EV3_KERNEL_util
{
        IN_8 src
        OUT_32 dst
```

```
        DATA8 tmp8
        MOVE8_8(src, tmp8)
        AND8(tmp8, 127, tmp8)
        MOVE8_32(tmp8, dst)
        JR_GTEQ8(src, 0, end_label)
        OR32(dst, 128, dst)
        end_label:
}

// utils functions block end
```

Внешний вид робота




EV3